UNIVERSITÀ DEGLI STUDI DI MILANO - FACOLTÀ DI SCIENZE

UMBERTO PELLEGRINI

TECNICHE DI ELETTRONICA DIGITALE

Volume 2 – Strutture dei sistemi per elaborazione dati

Edizioni La Viscontea

UNIVERSITÀ DEGLI STUDI DI MILANO - FACOLTÀ DI SCIENZE

UMBERTO PELLEGRINI

TECNICHE DI ELETTRONICA DIGITALE

Volume 2 — Strutture dei sistemi per elaborazione dati

Edizioni La Viscontea

PROPRIETÀ LETTERARIA RISERVATA

1979

Grafiche G. V. - Milano

Capitolo 1

EVOLUZIONE DEI SISTEMI DI ELABORAZIONE DI DATI

1.1. LA TECNOLOGIA ELETTRONICA E LE GENERAZIONI DEI CALCOLATORI

L'attuale sviluppo dei calcolatori è stato influenzato da tre fattori dominanti:

- le tecnologie, cioè i componenti con cui il calcolatore è costruito;
- le tecniche di progettazione circuitale e di programmazione, cioè il modo di comporre in strutture organizzate o sistemi i componenti disponibili;
- le applicazioni, cioè le possibilità d'uso dei sistemi.

Le tecnologie elettroniche hanno giocato il ruolo predominante tanto che, per classificare le successive "generazioni" dei calcolatori, si prende come parametro caratterizzante, il tipo di tecnologia usata.

L'introduzione dei componenti discreti a stato solido ha segnato verso la fine degli anni 50, il passaggio dalla prima generazione a valvole alla seconda generazione a transistori. Intorno alla metà degli anni sessanta l'uso dei circuiti integrati ha dato l'avvio alla terza generazione. All'inizio degli anni settanta assistiamo all'ulteriore evoluzione che dà origine alla cosiddetta "quarta generazione" determinata dalla microelettronica con le tecnologie LSI (Large Scale Integration).

Da una generazione alla successiva è aumentata via via l'esigenza di attuare i nuovi sistemi con un lavoro di sviluppo sempre più integrato su competenze tecnologiche estese in modo interdisciplinare, come mostrato nella Tabella 1.1.

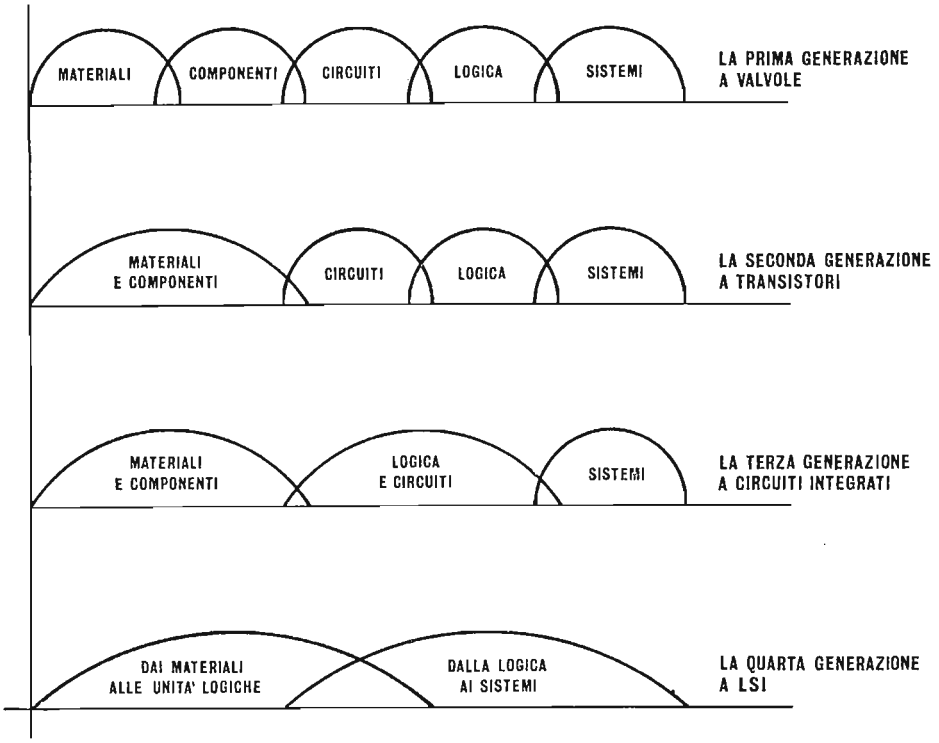


TABELLA 1.1 – L'EVOLUZIONE DELLE GENERAZIONI DI CALCOLATORI E L'ATTIVITA' DI SVILUPPO IN SENSO INTERDISCIPLINARE

Infatti, nel periodo dei componenti discreti, l'industria era organizzata su diverse categorie di fabbricanti: materiali, componenti, strumenti, sistemi. Le tecniche di integrazione dei componenti elettronici tendono a ridurre queste categorie spingendo le aziende a classificarsi in due grandi filoni complementari: il primo comprendente le aziende che dal materiale estendono oggi la loro attività fino alla produzione di dispositivi integrati funzionali o unità logiche operative; il secondo formato da aziende che usano le suddette unità logiche per attuare non solo gli strumenti ma anche il progetto, la realizzazione e l'installazione di sistemi di automazione.

Sicché dalla prima alla seconda generazione dei calcolatori abbiamo visto che lo sviluppo dei componenti a stato solido si è fuso con quello dei materiali. Nella terza, non è stato più possibile separare le prestazioni circuitali della famiglia dei componenti integrati usati dal progetto delle unità logiche. Con l'avvento della quarta generazione appare sempre più evidente come il risultato finale sia frutto di una stretta cooperazione interdisciplinare che mette insieme da una parte i ricercatori della tecnologia dei materiali con i progettisti dei circuiti e delle unità logiche LSI, e dall'altra ancora questi stessi progettisti con gli analisti di sistema e gli specialisti delle applicazioni.

Per quanto riguarda le prestazioni, abbiamo visto che le prime generazioni sono state sviluppate tenendo presente soprattutto l'obiettivo del calcolo automatico, migliorando via via le prestazioni "hardware" ed in primo luogo la velocità. Con la terza generazione l'accento si è decisamente spostato più in generale sull'acquisizione ed elaborazione dei dati (EDP) e sulla sistemistica, sicché abbiamo visto perfezionare le tecniche a divisione di tempo (time-sharing) e le tecniche di multiprogrammazione.

Oggi l'utilizzatore è interessato alle prestazioni ed al costo del sistema totale piuttosto che a quelle della sola unità centrale, la quale costituisce ormai una parte minoritaria (dal 20% al 30%) di tutto l'hardware. Perciò nei sistemi della quarta generazione, i criteri di progetto sia dell'hardware che del software, tendono a ottimizzare le tecniche ed i metodi di interconnessione e di comunicazione fra le unità del sistema e fra sistemi diversi.

La gestione del sistema sarà sempre più controllata dal flusso dei dati e dalle interfacce per la comunicazione dei dati più che dai programmi memorizzati, come avveniva nelle macchine precedenti. I dati delle diverse entrate verranno accettati ed elaborati entro i tempi di risposta imposti o desiderati. Le interazioni fra i dati ed i programmi saranno studiate soprattutto in vista delle risposte a ciclo chiuso di sistemi operanti in tempo reale.

1.2. LA MICROELETTRONICA E LE PRESTAZIONI DEI NUOVI SISTEMI.

Le tecnologie LSI, per integrazione a larga scala, hanno subito un tale sviluppo applicativo che da curiosità di laboratorio, nel giro di pochi anni, si

sono affermate ormai come un nuovo determinante fatto industriale.
Le tecniche LSI dei transistori bipolari a giunzione, che raggiungono su un singolo substrato l'integrazione fino a diverse migliaia di componenti, sono ormai largamente superate dalle tecniche dei transistori unipolari ad effetto di campo MOSFET-LSI (Metal Oxide Semiconductor Field Effect Transistor) che consentono di raggruppare su un'area equivalente sino a diverse centinaia di migliaia di elementi.
In realtà, se la riduzione di dimensioni è un fatto caratteristico della microelettronica, ancora più innovativo è il processo di **fabbricazione a lotti** – "batch processing" –, che è al tempo stesso causa ed effetto della miniaturizzazione raggiungibile. Quando si scende al di sotto di certe dimensioni fisiche, l'impossibilità pratica di manipolare i componenti uno alla volta con i procedimenti tradizionali, impone l'adozione di processi di costruzione automatizzata di tipo collettivo a molti componenti i quali, oltre a permettere il salto di scala nelle dimensioni geometriche, conseguono altri eccezionali vantaggi sia funzionali che economici elencabili come segue.

La miniaturizzazione consente di raggiungere nei circuiti **velocità operative maggiori**, proprio perché vi è una drastica riduzione dei percorsi geometrici e quindi dei tempi di propagazione dei segnali. Questa caratteristica può essere valutata nei giusti termini quando si tiene conto che la velocità con cui oggi operano i circuiti logici comporta tempi di commutazione nel campo dei nanosecondi, e che in un nanosecondo un'onda elettromagnetica percorre nell'aria non più di 30 cm.
Non sarebbero perciò possibili le attuali velocità senza una riduzione delle dimensioni.

Un altro parametro favorevolmente influenzato dalla microelettronica, specialmente con le tecniche COSMOS (Complementary-Semiconductor-MOS), è la **dissipazione di energia** in conseguenza sia dell'ottimizzazione dei carichi sia della possibilità di attuare rami fra tensione di alimentazione e massa con due interruttori in serie comandati dallo stesso segnale in modo che quando uno è in chiusura l'altro si trovi in apertura e viceversa: ciò consente ovviamente di raggiungere dissipazione nulla in condizioni stazionarie con un contributo di dissipazione trascurabile dovuto alle sole fasi di commutazione.

Ma la fabbricazione a lotti dei componenti LSI ha un effetto ancor più determinante sulla **affidabilità** dei circuiti per diversi motivi: da una parte il processo richiede un intervento umano trascurabile nella fase esecutiva;

dall'altra esso si svolge con procedure standardizzate e in ambiente controllato; la stessa minore dissipazione di energia riduce anche la degradazione nel tempo dei componenti; infine le saldature, che costituiscono le operazioni più critiche per l'affidabilità del circuito, vengono ridotte drasticamente di numero.

La maggior integrazione, l'automazione del processo ed i rendimenti più elevati raggiungibili con la fabbricazione a lotti, comportano anche una **diminuzione dei costi** di produzione. A questo riguardo si deve tener presente che il risparmio globale non è solo dovuto alla riduzione di costo del circuito corrispondente all'unità funzionale LSI, ma anche ad altri fattori che influenzano il costo del sistema totale quali: il minor numero di interconnessioni, la minore potenza di alimentazione, e le strutture di assemblaggio ridotte e semplificate.

Se noi consideriamo ad esempio l'unità centrale di un calcolatore, la cosiddetta CPU - *Central Processing Unit*, e prendiamo a riferimento una unità con 10.000 elementi attivi, possiamo constatare la drammatica riduzione di volume e di prezzo dovuta alle nuove tecnologie dai dati riportati nella Tabella 1.2.

TABELLA 1.2 - EVOLUZIONE DI UN SISTEMA DI ELABORAZIONE

Tecnologia hardware	Numero componenti	Volume m^3	Prezzo $
- a valvole (1955)	10.000	20	10^6
- a transistori (1965)	10.000	2	10^5
- a LSI (1975)	10.000	10^{-7}	10^2
- a LSI operazionali (1980)	500.000	10^{-7}	10^2

1.3. I CRITERI DI PROGETTO DEI NUOVI SISTEMI CON I COMPONENTI DELLA MICROELETTRONICA.

La tecnica LSI conduce a realizzare su singole piastrine grandi matrici di componenti, collegati in modo da formare unità funzionali a livello di sottosistema.

Ponendosi perciò a considerare le cose dal punto di vista del progettista di sistema sorgono nuovi problemi come conseguenza del fatto che al crescere della complessità dei circuiti integrati, pensati come unità funzionali, i componenti elettronici di cui il sistema è composto tendono ad essere meno ripetitivi e più diversi. Cambiano perciò i criteri di impostazione e di ottimizzazione del progetto. Ad esempio il concetto base per cui nel progetto di circuiti logici a componenti discreti o integrati è bene **minimizzare il "conto dei circuiti"** va completamente rinnovato giacché, con le tecniche LSI, l'impostazione è piuttosto quella di **minimizzare le connessioni**, anche a costo di uno spreco di circuiti.

Questo tipo di impostazione si riflette a sua volta nel progetto per definire i **parametri delle matrici LSI** quali: il numero o le dimensioni delle celle, la complessità o numero di gates per cella, il numero di terminali di interconnessione per cella, la dissipazione, le condizioni topologiche che rendono possibile una programmazione automatica delle interconnessioni fra celle per differenti funzioni logiche.

Si ottengono così matrici "multifunction" che rendono possibili nuove organizzazioni dei sistemi anche per quanto riguarda i compiti affidati allo "hardware" e al "software". La suddivisione delle rispettive aree va modificandosi rapidamente giacché, se in passato si assisteva ad un continuo spostamento di compiti dall'hardware al software, oggi si osserva una parziale inversione di tendenza con il cosiddetto "**firmware**". Infatti da una parte le tecnologie con matrici LSI e dall'altra le tecniche di microprogrammazione consentono di attuare unità di sottosistema che vengono progettate con criteri software e realizzate con tecniche hardware, per cui costituiscono in pratica un patrimonio **fisso**, - da cui il termine firmware - di sottoprogrammi del sistema.

Nei sistemi in sviluppo si potrà così affidare ad una singola unità LSI l'esecuzione di subroutines, la generazione di caratteri, la conversione di codici, l'impaginazione di tabelle di consultazione. Sarà agevole organizzare elaborazioni in parallelo. Le matrici usate come memorie di trasferimento potranno assumere, temporaneamente, anche la funzione di memorie di Controllo e di lavoro. Infine con le unità LSI vi è la concreta possibilità di introdurre nei sistemi un elevato **grado di ridondanza** a livello delle unità funzionali in modo da consentire capacità di autodiagnostica e di autoriparazione alle parti del sistema.

1.4. EVOLUZIONI E TENDENZE DEI SISTEMI DI ELABORAZIONE DI DATI.

Il sistema di elaborazione centralizzato, il cosiddetto **general purpose computer**, è stato alla base degli sviluppi dell'informatica negli anni '60: la sua organizzazione era concepita in modo da dare una procedura generale ed universale di soluzione alle richieste di servizio e alle applicazioni più diverse. La tecnologia di allora consentiva di ritenere ancora valida la cosiddetta legge di Grosch secondo cui all'aumentare delle dimensioni di un sistema di elaborazione centralizzato, le prestazioni aumentano in misura superiore rispetto all'aumento di prezzo.

Questa situazione viene oggi capovolta per diversi motivi. Il primo è che il pretendere di applicare procedure universali ai problemi più diversi conduce a sistemi di elaborazione ed a metodologie di programmazione che mettono sempre più in evidenza due fenomeni ben noti ma diventati più vistosi e limitativi con l'aumentare delle dimensioni dei sistemi:

– il costo crescente del software rispetto all'hardware, che è variato nel tempo come mostrato in Tabella 1.3.

TENDENZA NEL RAPPORTO DI COSTO HARDWARE-SOFWARE

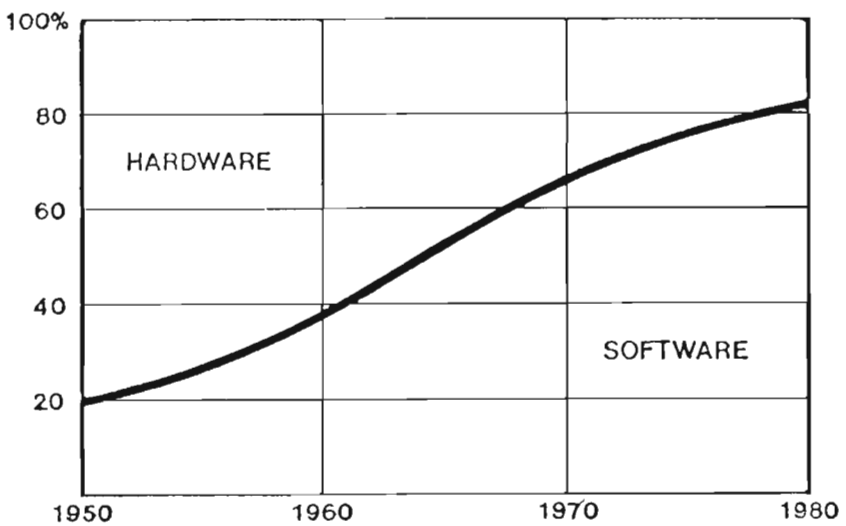


TABELLA 1.3

Si vede infatti che la situazione si è rovesciata nel giro di venti anni: l'hardware che negli anni '50 costituiva circa l'80 per cento del sistema, oggi scende al 20 per cento e l'opposto si verifica per il software.
Questa tendenza, è causata dal fatto che i dispositivi LSI introducono in tutte le apparecchiature capacità di elaborazione, di memoria e di comunicazione dati. La superiorità che i nuovi sistemi vengono così a conseguire rispetto ai precedenti va ricercata nel fatto che in essi le operazioni da eseguire non sono prestabilite e limitate dallo schema circuitale adottato, ma possono essere **programmate** a seconda dell'applicazione desiderata: il programma essendo di volta in volta immagazzinabile in memoria.

— l'efficienza decrescente del software di sistema, cioè dei programmi che gestiscono le risorse del sistema, al crescere della complessità del sistema centralizzato.

Tutto questo può essere commentato dicendo che anche per i calcolatori vale la legge di Parkinson relativa alle grandi organizzazioni centralizzate: oltre certe dimensioni del sistema di elaborazione, l'organizzazione impegna risorse proporzionalmente maggiori per controllare e gestire se stessa, sicché il software operativo che la governa diventa sempre meno efficiente.

Queste considerazioni, insieme a quanto è stato detto circa le prestazioni dei dispositivi funzionali LSI che su un solo componente raggiungono oggi la capacità di elaborazione di un medio calcolatore di dieci anni fa, sono alla base della vera rivoluzione alla quale assistiamo oggi nel campo dell'informatica che dai sistemi centralizzati va verso i sistemi distribuiti: unità di elaborazione, orientate verso applicazioni specializzate, sono localizzate direttamente presso l'utente e sono dotate di capacità per trasmissione dati in modo da poter essere collegate ad altre unità vicine o lontane sì da costituire reti di sistemi distribuiti.

Le singole unità sono organizzate in modo autonomo per la soluzione di problemi di specifici settori applicativi e nello stesso tempo sono in grado di comunicare con altre unità per rendere le varie risorse mutuamente disponibili.

Le proprietà che caratterizzano le unità di elaborazione dei sistemi distribuiti sono perciò le tre seguenti:

— ogni unità è localizzata presso l'utente ed è operata direttamente dallo stesso utente;
— ogni unità è dedicata ad una singola area applicativa o ad un gruppo di applicazioni simili;

— ogni unità può scambiare informazioni con altre unità attraverso le reti di trasmissione dati.

Una struttura di sistemi distribuiti così concepibile ottimizza anche i costi di trasmissione dati, al contrario di quanto potrebbe sembrare a prima vista, ed è perciò in armonia con quanto suggerito dalla Tabella 1.4. Infatti, mentre nei sistemi centralizzati collegati a distanza in time-sharing con terminali non-intelligenti, si dovevano trasmettere al centro tutti i dati raccolti in periferia, viceversa nei sistemi distribuiti, dove le elaborazioni avvengono localmente, la quantità di dati da trasmettere è molto minore perché si scambiano solo i dati più significativi già sottoposti ad elaborazione e a selezione.

COSTO PER MUOVERE 1 MILIONE DI BITS
SU UNA RETE DI CALCOLATORI
PER COMUNICAZIONE DATI

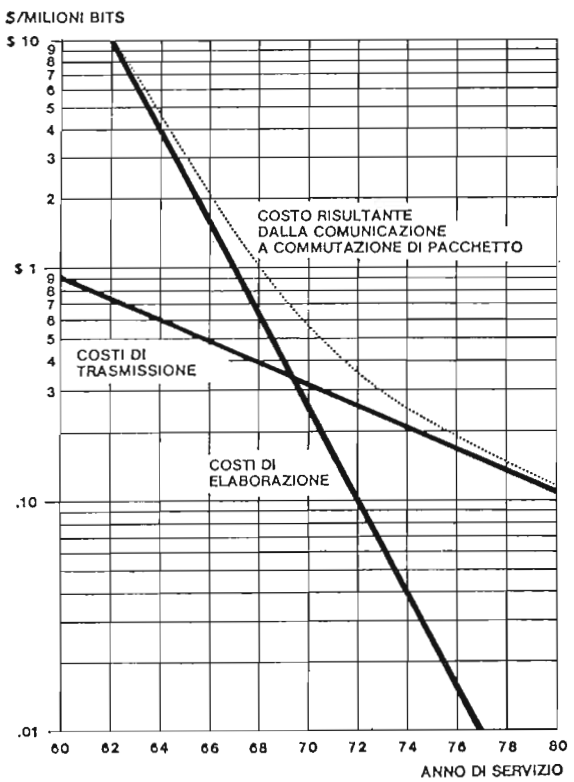


TABELLA 1.4

Come conseguenza di tutte queste trasformazioni tecnologiche anche la struttura interna del mercato dell'informatica si modifica secondo i dati della Tabella 1.5 ricavati da uno studio di mercato dello Stanford Research Institute.

I grandi sistemi centralizzati che all'inizio degli anni '70 coprivano oltre il 70% del mercato continueranno ad avere un incremento annuo del 5%, ma in percentuale la loro incidenza gradualmente si ridurrà perché il mercato delle unità dell'informatica distribuita aumenterà ad un tasso compreso tra il 25% ed il 30% annuo, sicché nel 1985 la situazione si sarà praticamente rovesciata.

COMPOSIZIONE PERCENTUALE MERCATO MONDIALE INFORMATICA
(esclusi i sistemi non business)

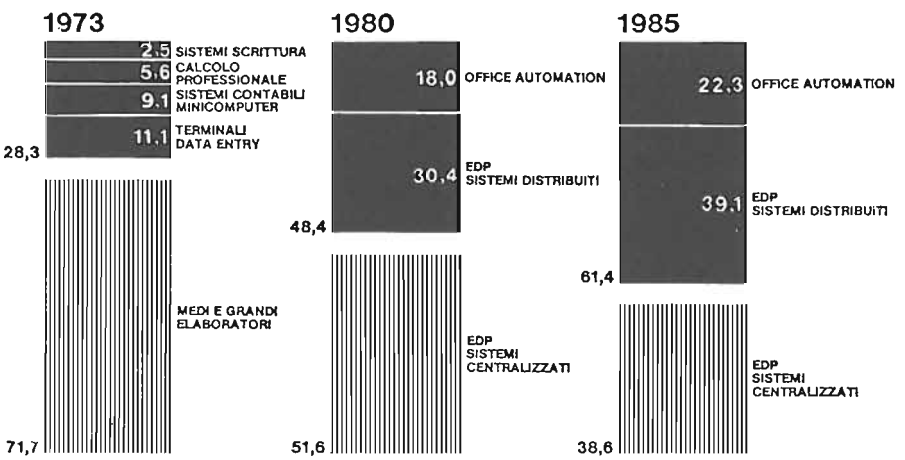


TABELLA 1.5

Capitolo 2

STRUTTURE HARDWARE.

2.1. STRUTTURA HARDWARE DI UN SISTEMA MINICOMPUTER.

Un tipico sistema con minicomputer è composto da quattro blocchi funzionali, ognuno destinato a svolgere un compito specifico, così come è schematizzato in fig. 2.1.

La **unità centrale di elaborazione** CPU - Central Processor Unit - controlla il funzionamento del sistema e compie tutte le operazioni logiche ed aritmetiche.

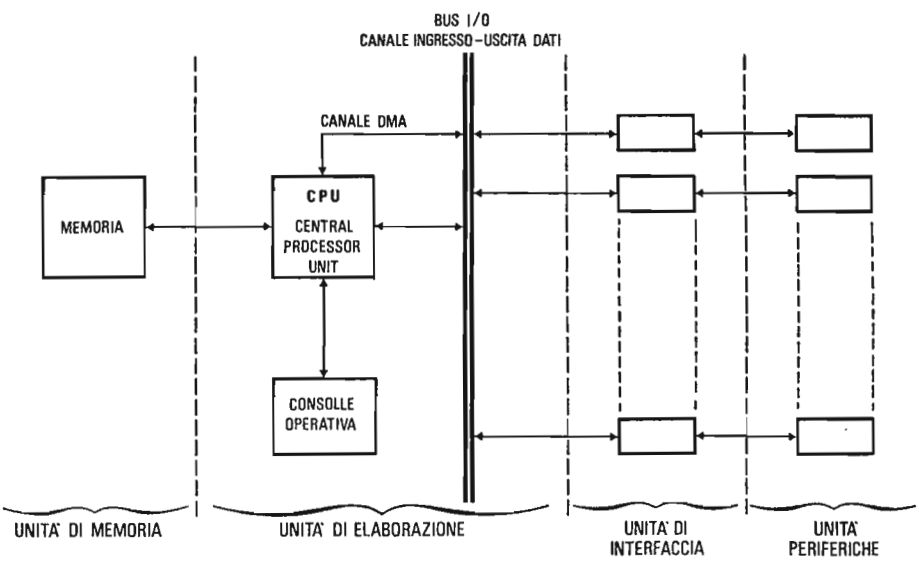


Fig. 2.1 -
SCHEMA A BLOCCHI CHE DESCRIVE L'ORGANIZZAZIONE A SISTEMA DI UN MINICOMPUTER CON UNITA' PERIFERICHE.

La **unità di memoria** è destinata ad immagazzinare i dati ed i programmi che contengono le istruzioni per le operazioni del CPU.

Le **unità periferiche** di ingresso e di uscita dei dati, come convertitori analogici-numerici, multiplexers digitali, visualizzatori, dischi e registratori magnetici, perforatori e lettori di nastro o di schede, stampatrici e telescriventi, sono collegate ciascuna alla CPU tramite una **unità di interfaccia** che adatta le connessioni elettriche, i livelli dei segnali ed i codici nella forma compatibile per il trasferimento dei dati con la CPU.

La CPU può comunicare con le unità periferiche in due modi: attraverso il "**bus di ingresso-uscita**" che è costituito da un insieme di fili di connessione tra le unità di interfaccia ed i registri accumulatori della CPU; oppure attraverso il canale o bus di **DMA - Direct Memory Access** che consente un trasferimento diretto con la memoria in tutti i casi in cui si ha a che fare con un flusso di dati ad alta velocità.

In generale nell'unità centrale del minicomputer si dispone anche di un pannello di controllo frontale che contiene lampade di visualizzazione, interruttori, e pulsanti per dare comandi dall'esterno e per controllare il funzionamento del calcolatore attraverso la presentazione dello stato di funzionamento dei diversi registri.

Un'unità periferica standard che si ritrova in tutti i sistemi di minicomputers, anche nella configurazione la più ridotta, è la telescrivente con tastiera per la stampa in chiaro e con perforatore e lettore di nastro o dischetto flessibile: essa costituisce il mezzo base per l'interazione e la comunicazione uomo-macchina.

2.2. STRUTTURA BASE DELLA CPU IN UN MINICOMPUTER.

L'unità centrale CPU di un minicomputer contiene tre sottounità operative come mostrato in fig. 2.2.:

- **l'unità di trasferimento dati con la memoria;**
- **l'unità di controllo** che governa sia il trasferimento dei dati entro la CPU sia la sequenza delle operazioni;
- **l'unità aritmetica** destinata a compiere tutte le operazioni aritmetiche e logiche sui dati.

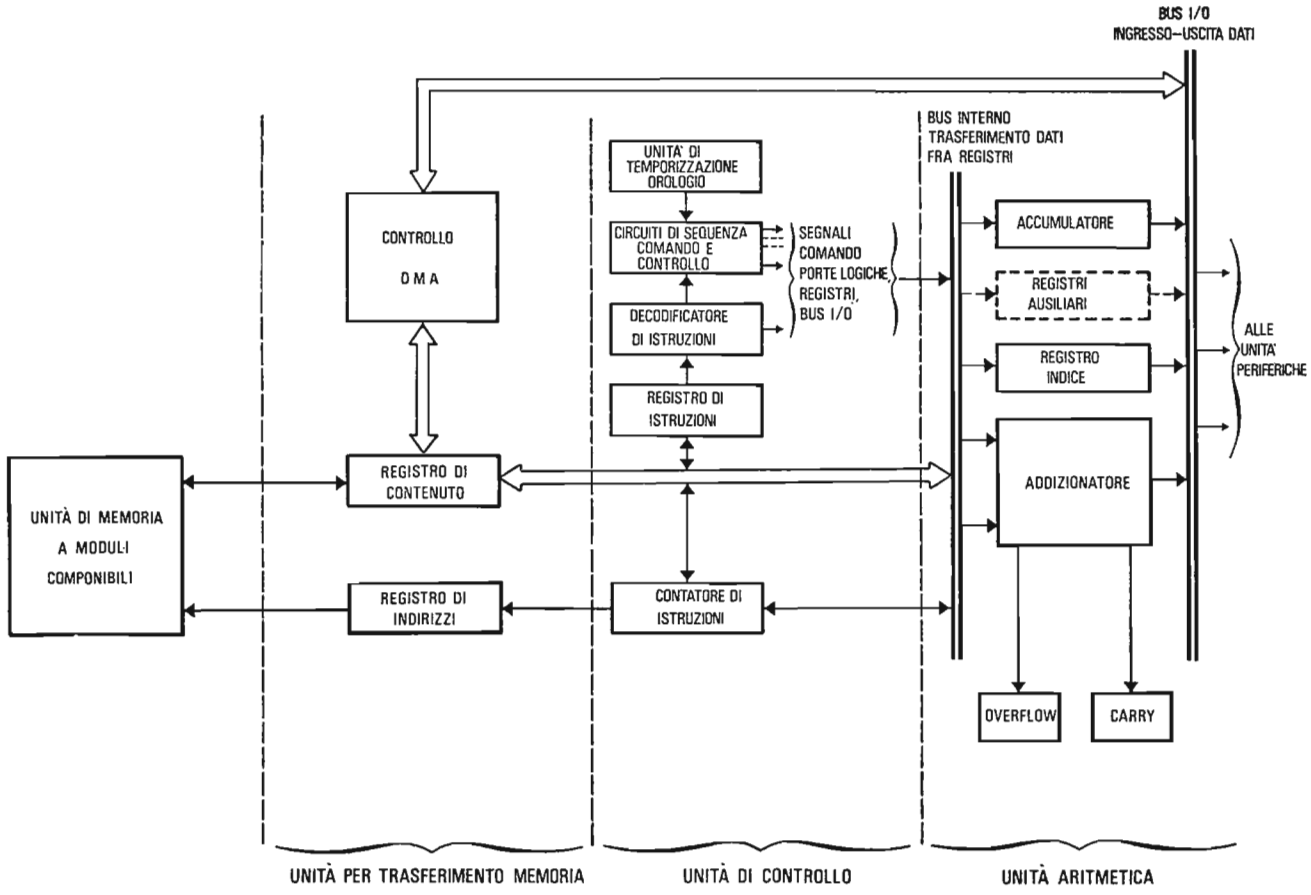


Fig. 2.2 – SCHEMA A BLOCCHI DELLE PRINCIPALI UNITÀ FUNZIONALI DELLA CPU.

Ciascuna di queste unità è formata da un certo numero di registri collegati come descritto in figura. Il numero totale dei registri varia da un modello all'altro, ma tutti i minicomputers contengono i registri base indicati nello schema.

Il funzionamento della CPU può essere descritto in modo sintetico riferendosi alle funzioni affidate ai registri costituenti ciascuna delle tre unità anzidette.

L'unità per il trasferimento con la memoria contiene sempre due registri base:

- **il registro di contenuto** usato come memoria temporanea per il contenuto di parola in tutti i trasferimenti alla o dalla memoria; ogni dato o istruzione passa sempre attraverso questo registro sia nell'entrare che nell'uscire dalla memoria;
- **il registro di indirizzo** usato per definire l'indirizzo della locazione di memoria in cui si vuole introdurre o da cui si vuole estrarre l'informazione che viene temporaneamente memorizzata nel registro di contenuto.

L'unità di controllo contiene principalmente due registri e i circuiti di temporizzazione-sequenza e decodifica con le seguenti funzioni:

- **il registro di istruzione** in cui si immagazzina l'istruzione in corso di esecuzione. L'istruzione viene estratta dalla memoria, trasferita in questo registro e quindi decodificata. La decodificazione fornisce i vari segnali necessari a comandare la sequenza delle operazioni richieste per l'esecuzione dell'istruzione;
- **il contatore di istruzioni** in cui si mantiene l'indirizzo di memoria della prossima istruzione da eseguire. Sicchè alla fine di ogni ciclo di istruzione, questo contatore viene automaticamente incrementato di una unità. E' quindi questo contatore che stabilisce quando un indirizzo, inviato al registro di indirizzi, è relativo ad un dato oppure ad una istruzione;
- **circuiti di temporizzazione, di sequenza e di decodifica** destinati a:
  1) fornire i segnali di temporizzazione in modo che ogni unità compia l'operazione programmata al tempo dovuto;
  2) generare la sequenza degli stati per i circuiti logici associati alle operazioni da svolgere;

3) decodificare le istruzioni in opportuni segnali per comandare le porte logiche interessate alle operazioni in corso.

L'unità aritmetica può contenere un numero più o meno elevato di registri, in ogni caso essa è costituita almeno dai seguenti:

- **unità addizionatore** che esegue tutte le operazioni aritmetiche (addizione, sottrazione, ecc.) e le operazioni logiche (AND, OR, OR-Esclusivo, ecc.);
- **registro accumulatore** che ha un impiego di uso generale per l'esecuzione della maggior parte delle istruzioni. Normalmente può costituire uno degli ingressi per l'addizionatore sia nelle operazioni aritmetiche che logiche; può essere impiegato nei trasferimenti dei dati con le unità d'ingresso e uscita; oppure può essere usato per confrontarne il proprio contenuto con quello di una locazione di memoria;
- **registro indice** che viene usato nelle operazioni di indirizzamento della memoria per modificare e ottenere nuovi indirizzi durante lo svolgimento di un programma;
- insieme con i registri precedenti, tutti aventi lo stesso numero di bits pari alla lunghezza di parola adottata della memoria del calcolatore, vi sono spesso uno o due registri ciascuno ad un solo bit destinati a segnalare quando si superano le capacità aritmetiche o logiche dell'accumulatore. Questi registri sono chiamati:
  - **registro di overflow** ad 1 bit impiegato per segnalare un superamento delle capacità aritmetiche dell'addizionatore. Poichè nella presentazione dei numeri generalmente adottata nei minicomputers, il bit più significativo della parola viene usato per indicare il segno (in modo che un numero di **n** bits possa esprimere valori dal più piccolo numero negativo -2^{n-1} al più grande positivo $+2^{n-1}-1$), il superamento della capacità aritmetica dell'addizionatore si ha quando un'operazione aritmetica determina un riporto **sul bit più significativo 2^n della parola dell'addizionatore.** Ad esempio, nel caso di una parola di 8 bits, cioè di 1 byte (*), la somma sottoindicata dà un riporto sul bit più significativo e perciò porta ad 1 il registro di overflow:

(*) un BYTE è una sequenza di 8 bits considerati come un insieme.

```
 0 | 1 0 1 1 1 1 1
 0 | 0 1 0 1 1 1 1
 ---------------------
 1 | 0 0 0 1 1 1 0
 ↑
 └──── bit di overflow
```

– **registro carry** (di riporto, detto anche extended o di estensione) ad 1 bit impiegato per segnalare un superamento delle capacità logiche dell'addizionatore. Esso viene messo ad 1 ogni qualvolta un'operazione aritmetica determina un riporto **dal bit più significativo della parola dell'addizionatore**. Cioè si deve avere in questo caso un riporto dal bit 2^n e non sul bit 2^n che nella presentazione aritmetica viene usato per indicare il segno. Ad esempio si ha un riporto **carry o extended** nella somma sottoindicata dove si suppone la parola sempre pari a un byte:

```
   0 | 1 0 1 1 1 1 1
   1 | 0 1 0 1 1 1 1
 ---------------------
1  0 | 0 0 0 1 1 1 1
↑
└──── bit carry o extended
```

L'unità centrale CPU comunica con l'esterno attraverso il **BUS o Canale I/O (Input-Output) di ingresso-uscita dati.**

Il **canale I/O** viene spesso chiamato con il nome di BUS per indicare che è organizzato su linee multiple in parallelo in modo da permettere il trasferimento di più dati simultaneamente.
Il BUS I/O costituisce la via normale per collegare una periferica di ingresso o di uscita che viene gestita dal programma operativo, residente in memoria, attraverso l'unità di interfaccia.

Il **canale DMA - Direct Memory Access**, come indica il nome, è usato invece per trasferire i dati direttamente dalla o alla memoria con la massima velocità consentita dall'unità di memoria stessa, che in genere è più elevata di quella possibile sul canale di input-output programmato.
Il modo di operare del canale I/O verrà descritto più dettagliatamente in seguito dopo aver illustrato la struttura delle istruzioni e la funzione di temporizzazione dei cicli operativi della macchina regolati dall'unità di controllo.

2.3. STRUTTURE DELLE ISTRUZIONI DI PROGRAMMA.

E' ovvio che per elaborare informazioni, il calcolatore deve poter compiere un insieme di operazioni base sia di trasferimento dei dati fra registri sia di calcolo sui dati contenuti negli stessi.

Per svolgere una complessa elaborazione noi dobbiamo quindi sviluppare sempre **un algoritmo - cioè una lista di istruzioni che descrive come detta elaborazione può essere effettuata a passi successivi** - usando solamente quelle operazioni che possono essere eseguite dalla CPU.

Questa lista o sequenza di istruzioni è ciò che viene denominato un **programma**.

Per capire come la CPU esegue il programma dobbiamo perciò **capire come le parole istruzioni, che descrivono le operazioni da effettuare in ogni passo del programma, sono codificate** e come esse sono usate dall'unità di controllo della CPU.

Vi sono tre classi generali di istruzioni:

- istruzioni di riferimento alla memoria;
- istruzioni di operazione dei registri;
- istruzioni di comando per l'ingresso-uscita.

Una parola d'istruzione è suddivisa in sezioni o campi che contengono informazioni di tipo diverso: questi campi sono sostanzialmente costituiti dall'insieme di un certo numero di bit adiacenti nella parola a cui si affida un particolare tipo di significato. Il modo di sezionare la parola in campi e il codice dato ad ogni campo per indicare una particolare operazione da svolgere stabilisce il **formato** della parola.

In termini molto generali si può affermare che una parola di istruzione contiene almeno due distinti campi come mostrato in fig. 2.3.

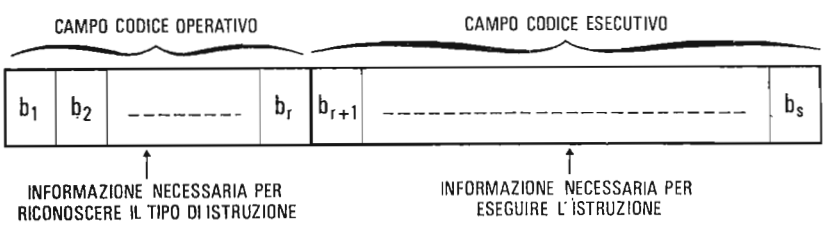


Fig. 2.3 -

E' chiaro che se il campo operativo ha r bits allora la CPU potrà disporre di tipi di istruzioni diverse pari a 2^r.

Le tre classi di istruzioni prima elencate possono avere formati differenti in funzione del tipo d'informazione che deve essere data all'unità di controllo.

Infine quasi ogni calcolatore ha le proprie istruzioni codificate in un formato diverso dagli altri.

2.3.1. Le istruzioni di riferimento alla memoria.

Queste istruzioni riguardano le operazioni relative all'informazione che è o va immagazzinata in una o più locazioni di memoria.

I compiti assegnati a queste istruzioni sono in genere i seguenti:

- leggere o scrivere i dati in memoria;
- compiere operazioni aritmetiche e logiche sui dati in memoria o nei registri;
- modificare la sequenza delle istruzioni secondo cui un programma viene svolto.

Sicchè una tipica istruzione o sequenza di istruzioni di riferimento memoria dovrebbe dare, ad esempio, le seguenti informazioni:
"compi una data operazione sull'operando A e sull'operando B, poni il risultato in una data locazione di memoria e procedi alla locazione di memoria che contiene l'istruzione successiva".

Ciò significa che occorre dare:

1 - il codice del tipo di operazione da svolgere;
2 - l'indirizzo in memoria del primo operando A;
3 - l'indirizzo in memoria del secondo operando B;
4 - l'indirizzo in memoria dove deve essere collocato il risultato;
5 - l'indirizzo in memoria dell'istruzione successiva.

Per questo i primi calcolatori e, a tutt'oggi, alcuni grandi calcolatori usano istruzioni cosiddette a quattro indirizzi, cioè sono divise in 5 campi, uno per il codice operativo e gli altri quattro per definire gli indirizzi come indicato in figura 2.4

Questo modo di codificare le istruzioni, pur essendo molto flessibile, presenta notevoli inconvenienti a causa dell'elevato numero di bits che richie-

CODICE OPERATIVO	INDIRIZZO I OPERANDO	INDIRIZZO II OPERANDO	INDIRIZZO RISULTATO	INDIRIZZO NUOVE ISTRUZIONI

Fig. 2.4 – ISTRUZIONI DI RIFERIMENTO MEMORIA A QUATTRO INDIRIZZI.

de nella parola di istruzione. Si pensi ad esempio che, anche nel caso di una memoria a sole $2^{12} = 4096$ parole, ogni indirizzo richiederebbe 12 bits, cioè solo per specificare gli indirizzi si richiederebbero 48 bits. Memorie con parole di questa lunghezza sono molto costose e fattibili solo per macchine di grandi dimensioni. Per questo nei successivi sviluppi dei calcolatori si sono studiate strutture semplificate con istruzioni a tre, a due indirizzi, fino ad arrivare nel caso dei minicomputers alle istruzioni ad un solo indirizzo.

Come è possibile modificare il formato delle istruzioni fino ad arrivare a quelle ad un solo indirizzo? Esaminiamo per questo come viene classificata oggi l'informazione relativa ad un generico programma immagazzinato in memoria. Le locazioni riservate alle parole di istruzione sono raggruppate in indirizzi successivi in una determinata sezione di memoria, mentre i dati da elaborare sono immagazzinati in una sezione diversa come mostrato in figura 2.5.

Se disponiamo del registro "**contatore di istruzioni**" già prima descritto, possiamo tenere traccia dell'indirizzo dell'istruzione che si sta eseguendo e conoscere automaticamente l'indirizzo della successiva senza doverla specificare nell'istruzione stessa. L'unica limitazione che introduce questo metodo è quella che siamo costretti ad introdurre istruzioni aggiuntive da usare quando si vuole o si deve alterare la sequenza delle istruzioni in cui il programma viene svolto. Dovendo effettuare operazioni su più operandi si predispone allora un'opportuna sequenza di istruzioni onde trasferire prima ogni operando dalla memoria in un opportuno registro della CPU, comandare l'operazione da eseguire e quindi trasferire il risultato dai registri alla memoria.

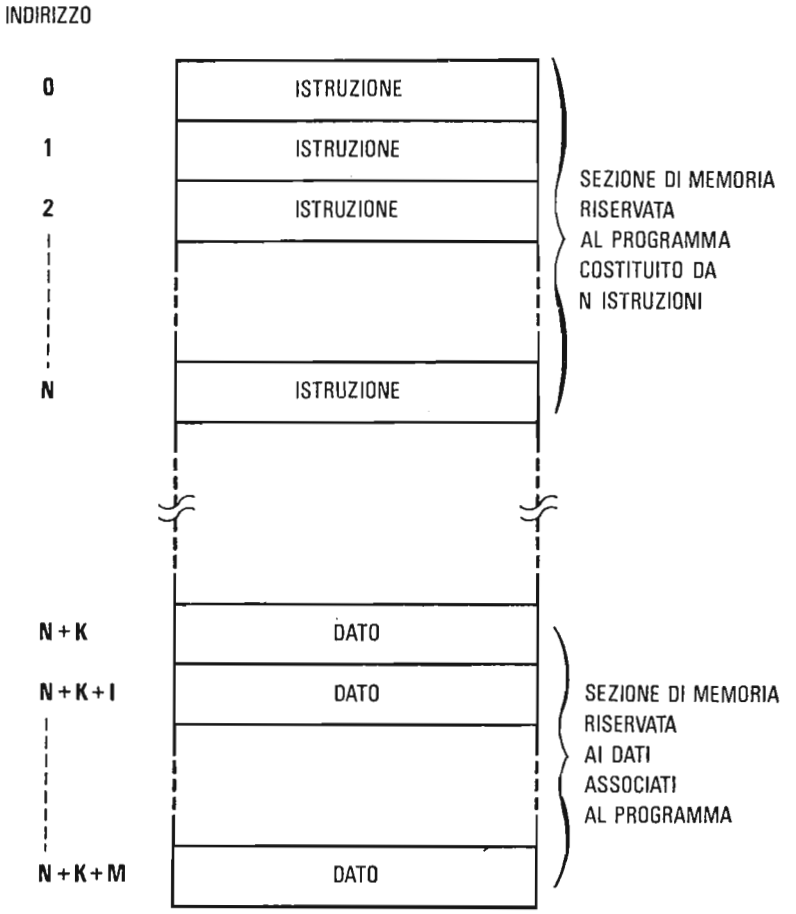


Fig. 2.5 -

2.3.2. Le istruzioni di riferimento in memoria ad un solo indirizzo nei minicomputers.

La maggior parte dei minicomputers ha la parola costituita da 16 o da 8 bits. Nel caso di parole a 16 bits il formato delle istruzioni di riferimento in memoria è in generale come mostrato in fig. 2.6.

Il campo con i bits 11-15 relativo al codice operativo specifica il tipo particolare di istruzione: ad esempio 10000 può significare " addiziona"; mentre 10100 equivale a "sottrai"; oppure 01001 indica "trasferisci nell'accu-

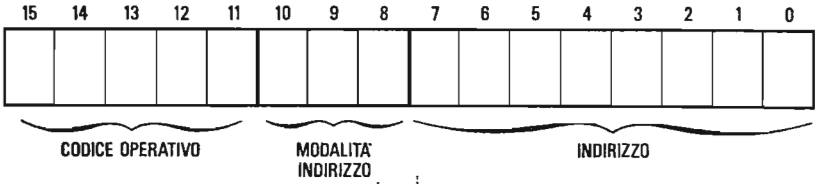


Fig. 2.6 -

mulatore" e così via. Se i bits riservati a questo campo sono 5 si potranno perciò avere $2^5 = 32$ istruzioni diverse. Il campo con i bits 0-10 viene generalmente suddiviso in due sottocampi: uno da 0 a 7 per specificare l'indirizzo di memoria, e l'altro da 8 a 10 per specificare la cosiddetta "**modalità d'indirizzamento**". Infatti, se il sottocampo contiene n bits, solo 2^n locazioni di memoria potrebbero essere specificate con un indirizzamento diretto usando gli n bits; ma se la memoria contiene più di 2^n locazioni occorre provvedere in modo diverso per tutti gli altri indirizzi.

Così con 8 bits, cioè con il codice del sottocampo dei bits 0-7, si possono specificare solo gli indirizzi di $2^8 = 256$ parole, mentre la memoria di lavoro di un minicomputer è composta di moduli di 4 K oppure di 8 K parole, e può raggiungere capacità fino a 64 K. Perciò per poter indirizzare qualsiasi locazione di memoria, si usano diverse modalità d'indirizzamento ognuna specificata con un codice nel sottocampo costituito dai bits 8-10.

Queste modalità, che possono variare leggermente da un calcolatore all'altro e che nel caso particolare del minicomputer LABEN 70 sono descritte a mo' di esempio nei paragrafi seguenti, si basano in generale sui seguenti metodi generali:

1 - **Metodo di indirizzamento indiretto** - in questo caso la locazione di memoria specificata dal codice d'indirizzo dei bits 0-7 (cioè l'indirizzo diretto) contiene l'indirizzo dell'operando piuttosto che l'operando stesso: sicchè in una macchina con parole da 16 bits si può indirizzare in modo indiretto qualsiasi locazione fino a $2^{16} = 64$ K indirizzi.
   Va notato che l'indirizzamento indiretto può essere a molti passi o, come si dice, in cascata: può cioè avvenire che si debba passare attraverso più di due posizioni di memoria prima di arrivare all'effettivo operando.

2 - **Indirizzamento (diretto o indiretto) relativo al programma** - in questo modo l'indirizzo dell'operando è ottenuto come somma dell'indirizzo contenuto nei bits 0-7 dell'istruzione **più il contenuto del registro contatore d'istruzioni**.
E' bene dire subito che in questo modo il formato dell'istruzione interpreta il valore dei bits 0-6 come indirizzo e usa il bit 7 come bit di segno più o meno.
Pertanto con tale modalità l'indirizzamento può variare da -128 a +127 locazioni situate intorno a quella in cui è contenuta l'istruzione (il cui indirizzo è specificato dal contatore d'istruzione).

3 - **Indirizzamento (diretto o indiretto) relativo al registro indice** - in questo modo l'indirizzo dell'operando è la somma dell'indirizzo contenuto nel sottocampo dei bits 0-7 dell'istruzione più il contenuto del registro indice. In una macchina con parole a 16 bits anche il registro indice sarà a 16 bits e si potrà perciò indirizzare ogni locazione di memoria. Questo modo, detto anche **indirizzamento indiciato**, è particolarmente vantaggioso in tutti i casi in cui si devono fare elaborazioni con operazioni ripetitive su una lista o su una matrice di dati ognuno classificato con uno o più indici a_{ij}. Si può così rendere più semplice la compilazione e quindi l'esecuzione del programma associando gli indici del dato ai contenuti dei registri indice (il minicomputer può disporre di più di un solo registro indice, oppure usare temporaneamente altri registri con la funzione di registro indice).

Per rendere questi modi di indirizzamento più flessibili alle esigenze della programmazione, **molti minicomputers dividono la memoria in pagine**: ogni pagina è costituita da 256 parole, tante cioè quante se ne possono indirizzare direttamente con i bits del sottocampo 0-7.

Il primo gruppo di 256 parole costituisce la pagina 0; il secondo gruppo da 257 a 512 costituisce la pagina 1 e così via.

Si può definire altresì la **pagina corrente** come la pagina contenente l'indirizzo specificato dal contatore di istruzioni: cioè l'indirizzo dell'istruzione che è in corso di esecuzione.

E' anche evidente come sia possibile spostare a programma l'inizio e la fine dei gruppi di 256 parole costituenti una pagina, in modo da ottenere **pagine mobili** entro la memoria piuttosto che **pagine fisse** come prima descritte.

2.3.3. Istruzioni di operazione dei registri.

Queste istruzioni servono per comandare le operazioni da compiere con i registri della CPU, quali ad esempio:

- predisporre il contenuto di un registro su valori predeterminati;
- trasferire o paragonare il contenuto di un registro con quello di un altro registro;
- aggiungere o sottrarre una costante al contenuto di un registro;
- controllare il contenuto di un registro per provare alcune proprietà logiche o aritmetiche;
- comandare scorrimenti o rotazioni del contenuto di un registro;
- alterare la sequenza del programma dando istruzioni alla unità di controllo di cercare la prossima istruzione in una diversa locazione da quella prima stabilita.

Queste ed altre operazioni analoghe sono individuate da istruzioni il cui formato si può presentare come mostrato in fig. 2.7.

Fig. 2.7 -

Il codice operativo 00000, non usato dalle istruzioni di riferimento memoria, specifica che si ha a che fare con una istruzione di operazione registri. I sottocampi dei bits da 0 a 10 che possono essere due o anche più, individuano la operazione da effettuare e gli indirizzi dei registri fra cui deve essere effettuata.

2.3.4. Istruzioni di comando per l'ingresso-uscita.

Questa terza classe d'istruzioni serve per trasferire l'informazione tra la CPU e le diverse unità periferiche. L'istruzione deve perciò contenere i seguenti comandi:

- selezionare la periferica a cui si riferisce l'istruzione;
- fornire l'informazione per eseguire la particolare funzione o tipo di trasferimento richiesto.

Il formato si può presentare perciò come mostrato in fig. 2.8.

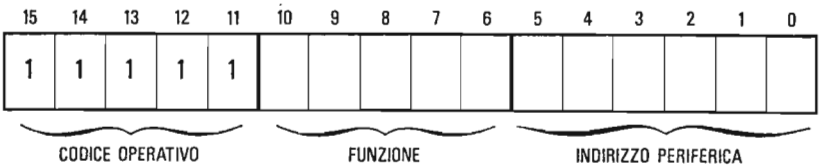


Fig. 2.8 -

Il codice operativo 11111, non usato dalle istruzioni di riferimento memoria, indica che l'istruzione è di ingresso-uscita.

Ogni periferica, individuata con un numero, è selezionata dal codice di sottocampo dei bits 0-5, mentre la funzione di trasferimento richiesta è specificata dal codice dei bits 6-10.

2.3.5. Istruzioni microprogrammate o microistruzioni.

Molti minicomputers adottano oggi anche una classe speciale di istruzioni dette "istruzioni microprogrammate o micro-istruzioni".

Ciò è dovuto al fatto che diverse operazioni da compiere nella CPU non richiedono codificazioni elaborate nè parole a molti bits.

Basta molte volte segnalare la presenza o l'assenza di un comando con un solo bistabile posto rispettivamente a 1 o a 0: ad esempio l'istruzione per effettuare il complemento di un determinato registro, o per incrementare di 1 il suo contenuto non richiede una parola completa ma il solo segnale di 0 o 1 di un bistabile. Queste ed altre simili operazioni elementari possono essere raggruppate insieme in una nuova classe di istruzioni caratterizzata da un proprio codice operativo e con un formato come mostrato in fig. 2.9

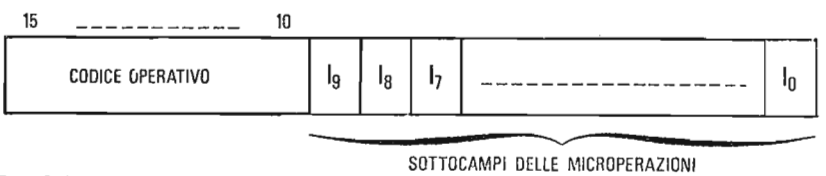


Fig. 2.9 -

Al solito si possono lasciare i bit 10-15 ad indicare il codice operativo, mentre tutti i restanti bits si dividono in sottocampi ciascuno ad un solo bit, ciascuno che corrisponde al comando di una particolare operazione.

Ad esempio I_0 può specificare l'istruzione di "incremento accumulatore", I_1 può significare "azzeramento accumulatore", I_2 può comandare "salta alla prossima istruzione se il contenuto dell'accumulatore è zero", e così via.

E' chiaro che se più microperazioni I_k sono poste in 1, contemporaneamente, si avrà l'esecuzione contemporanea durante un ciclo di tutte le corrispondenti operazioni. E' perciò necessario ridurre il numero delle possibili combinazioni di microperazioni, poiché alcune di esse non possono essere eseguite o possono essere in contraddizione.

L'estendersi dell'uso di queste "microistruzioni" nel programmare le varie operazioni della CPU nei minicomputers, ha fatto sviluppare molto le tecniche dei "micro-programmi" e le strutture di macchine con unità "microprogrammate". Una macchina si dice oggi microprogrammata se è fornita di un insieme di comandi elementari per compiere le varie operazioni della CPU. Questi comandi sono poi raccolti in una classe convenzionale di istruzioni le quali vengono memorizzate in una memoria a sola lettura ROM - Read-Only-Memory, raggiungendo diversi vantaggi.

Infatti la ROM ha un tempo di lettura molto inferiore a quello delle memorie di lavoro RAM, e ciò consente così un tempo di esecuzione delle microistruzioni più piccolo rispetto alle istruzioni convenzionali.

Infine nel microprogramma, l'insieme delle microistruzioni può essere adattato ai diversi tipi di applicazioni specializzate a cui è destinata la macchina. Si possono cioè impiegare memorie ROM intercambiabili ciascuna con un insieme di microistruzioni specializzate ma compatibili con le istruzioni generali della macchina usata, da inserire nella CPU a seconda della applicazione. Infine con l'aggiunta di ulteriori moduli ROM microprogrammati è possibile espandere ulteriormente sia i programmi standard che quelli specializzati.

2.4. L'UNITA' DI CONTROLLO E LA TEMPORIZZAZIONE DELLE FASI OPERATIVE DEL CALCOLATORE.

E' stato già detto che l'unità di controllo governa tutte le operazioni della CPU: oltre a fornire i necessari segnali per eseguire le istruzioni che mano

mano arrivano al registro di istruzioni durante lo svolgimento di un programma, essa deve anche controllare la sequenza temporale secondo cui tutte le operazioni vengono eseguite.

Sicché per questa funzione l'unità di controllo è una rete sequenziale che ha un certo numero di ingressi, che deve generare un certo numero di diversi segnali di comando in uscita e che procede nei successivi stati comandata dagli impulsi periodici di un orologio elettronico formato da un oscillatore controllato a quarzo.

Per orientarsi sulla funzione di temporizzazione svolta dall'unità di controllo, **supponiamo dapprima per semplicità di avere a che fare con un programma in cui tutte le istruzioni siano ad un solo indirizzo e con indirizzamento diretto.**

Assumiamo che le **m** istruzioni del programma siano memorizzate sequenzialmente in memoria nelle locazioni comprese dall'indirizzo **n** all'indirizzo **n + m**.

Per eseguire il programma l'unità di controllo legge la prima istruzione nell'indirizzo **n** e quindi inizia e completa le operazioni per eseguire detta istruzione. Quindi va all'indirizzo **n + 1** e analogamente esegue la seconda istruzione. Tale sequenza continua finchè s'incontra un'istruzione che comanda all'unità di controllo di bloccare le operazioni perché il programma è terminato oppure di andare in altra locazione dove trovare un'altra sequenza di istruzioni relativa ad un altro programma.
Da ciò risulta chiaro come all'unità di controllo siano affidate due funzioni che identificano le seguenti due diverse fasi operative dell'unità stessa:

- **fase di ricerca istruzione (fetch phase)**: in cui si deve ricercare l'istruzione desiderata seguendo una predeterminata sequenza;
- **fase di esecuzione istruzione (execute phase)**: in cui si deve procedere a decodificare l'istruzione e a fornire i necessari segnali per eseguirla correttamente.

Spesso però nella fase di ricerca istruzione l'indirizzo dell' operando non può essere completamente determinato, come già si è visto, dal codice contenuto nel solo campo di indirizzo dell'istruzione. In questo caso si distinguono i passi della sequenza necessaria per determinare l'indirizzo, identificandoli in una fase intermedia fra le due prima accennate, la quale viene chiamata:

– **fase d'indirizzamento indiretto (indirect phase)**: in cui di prolunga la fase di ricerca istruzione o indirizzandosi su nuove locazioni di memoria o modificando l'indirizzo con l'aiuto del registro indice.

Queste fasi si svolgono in un periodo di tempo compreso in uno o più "**cicli di macchina**". Il ciclo di macchina corrisponde ad un periodo di tempo fisso (generalmente dell'ordine del microsecondo), scandito dagli impulsi di un orologio elettronico (clock pulses) il quale in genere viene pilotato con un oscillatore a quarzo. La frequenza dell'oscillatore è scelta abbastanza alta (dell'ordine delle decine di MHz) rispetto a quella del ciclo di macchina in modo da garantire oltre alla stabilità di frequenza anche una sufficiente precisione nella durata del ciclo stesso.

Si può meglio chiarire le modalità di temporizzazione e la sequenza delle fasi operative in un minicomputer riferendosi, ad esempio, al calcolatore LABEN 70 ed alla sua struttura generale così come è riportata nello schema a blocchi fig. 2.10

L'unità di controllo del LABEN 70 può essere considerata come un circuito sequenziale totalmente sincrono i cui stati successivi sono avanzati da un orologio controllato a quarzo a 26,666 MHz, che genera impulsi di inizio ciclo ogni 1.35 μsec. Ad ogni ciclo di macchina viene iniziato generalmente anche un ciclo di memoria la cui durata effettiva è di 0.9 μsec e che è composto sia dal passo di lettura che di scrittura.

Ad ogni ciclo di macchina vengono comandate le operazioni elementari necessarie alla corretta esecuzione delle varie istruzioni relative alle fasi in via di svolgimento.

La sequenza delle diverse fasi non è rigida, ma determinata dalle condizioni che si verificano durante l'esecuzione dell'istruzione. Lo svolgimento di ogni fase può essere così descritta.

Fase di ricerca di istruzione: durante questa fase il contenuto della locazione di memoria, il cui indirizzo all'inizio del ciclo è nel registro L, viene letto ed immesso nel registro W, e quindi riscritto nella stessa locazione di memoria.
Tale contenuto viene trasferito quindi dal registro W al registro di istruzione I ed interpretato come codice di istruzione. Nel caso che l'istruzione sia di memoria, viene definito l'indirizzo dell'operando secondo le modalità indicate dal codice dell'istruzione.

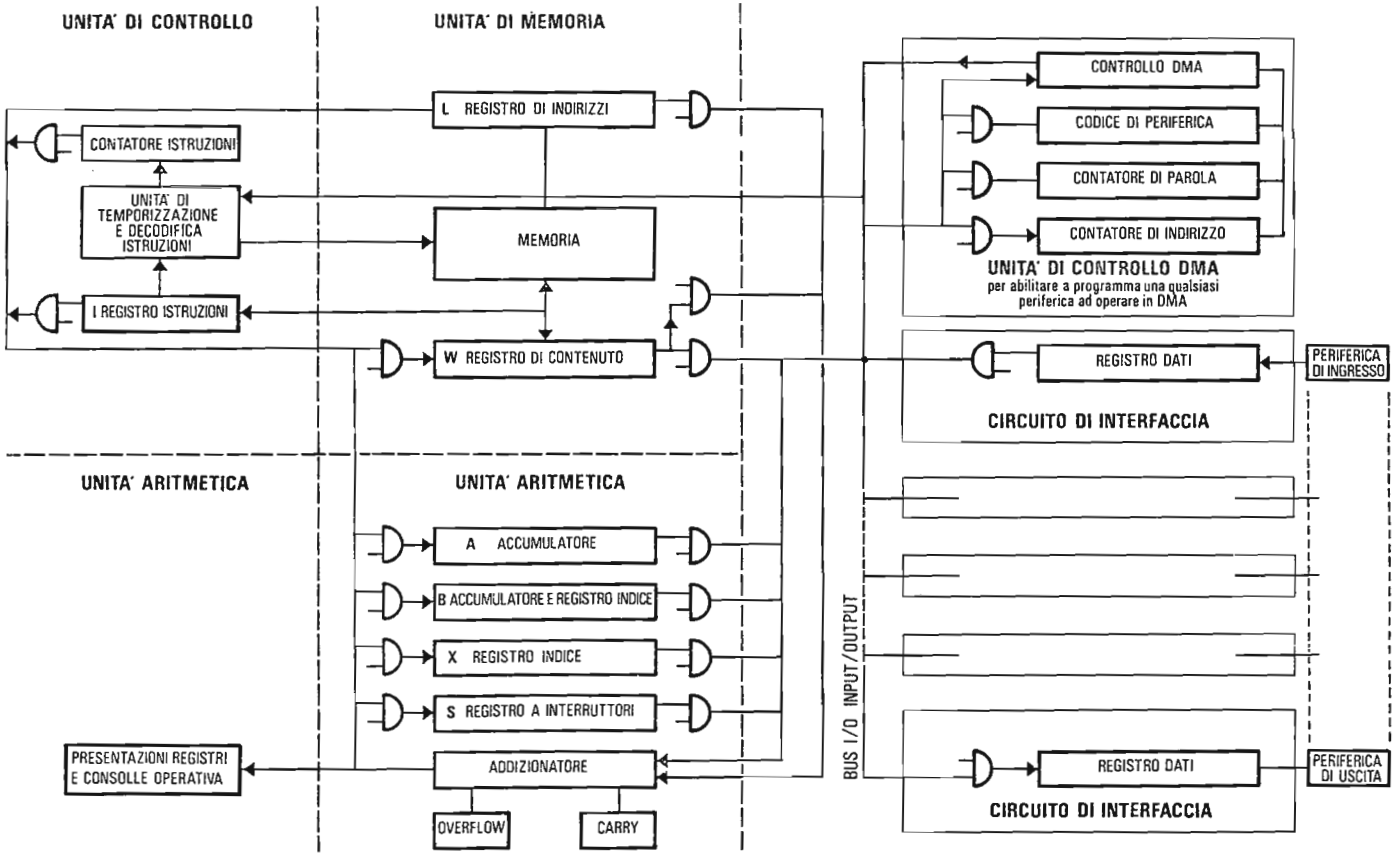


Fig. 2.10 - SCHEMA A BLOCCHI DELLE UNITA' FUNZIONALI DEL CALCOLATORE LABEN 70.

Alla fine della fase di ricerca di istruzione si possono verificare vari casi:

1 - se l'istruzione è del tipo che può essere totalmente eseguita in un solo ciclo, alla fine di tale fase viene impostato un nuovo ciclo di ricerca di istruzione e letta una nuova istruzione;
2 - se invece sono necessari altri cicli per l'esecuzione, viene impostata una fase di esecuzione;
3 - se il calcolo dell'indirizzo dell'operando non è completato alla fine del primo ciclo, si passa alla fase di indirizzamento.

Nella fase di indirizzamento si possono verificare questi casi:

— se il calcolo da completare per definire l'indirizzo richiede un **indirizzamento diretto con modifica da registro indice**, la fase di ricerca di istruzione viene prolungata per un altro ciclo, durante il quale non si ha ciclo di memoria, e viene invece completato il calcolo dell'indirizzo. A tale seconda fase di ricerca di istruzione può seguire una fase di esecuzione oppure, in casi particolari (istruzioni JMP, XEC), una nuova fase di ricerca con interpretazione di una nuova istruzione;
— se invece il calcolo dell'indirizzo dell'operando richiede un indirizzamento indiretto, alla fine della fase di ricerca di istruzione iniziale si imposta una fase di indirizzamento indiretto che si svolge come segue.

Fase di indirizzamento indiretto: durante questa fase viene letto il contenuto della locazione di memoria indirizzata e trasferito direttamente nel registro L. Si controlla il valore 0 o 1 del bit 15 del contenuto. Se questo bit vale 1, si procede ad un **nuovo** indirizzamento indiretto. Viene perciò impostata una nuova fase di indiretto e vengono ripetute le operazioni ora descritte, dando luogo ad una fase di indirizzamento indiretto a molti passi, cioè a molti cicli.

La catena di indiretti cessa allorché il bit 15 vale 0, specificando così che l'indirizzo contenuto nel registro L è l'indirizzo dell'operando.
Se il calcolo dell'indirizzo dell'operando richiede un **indirizzamento indiretto con modifica di registro** (X o B,X), viene prima impostata una fase di indiretto (eventualmente a molti passi), seguita da un ciclo durante il quale viene completato il calcolo dell'indirizzo con la modifica di registro X (o dei registri X e B).

Fase di esecuzione: durante questa fase, l'unità di controllo fornisce tutti i comandi necessari per realizzare l'esecuzione dell'istruzione. Le fasi di e-

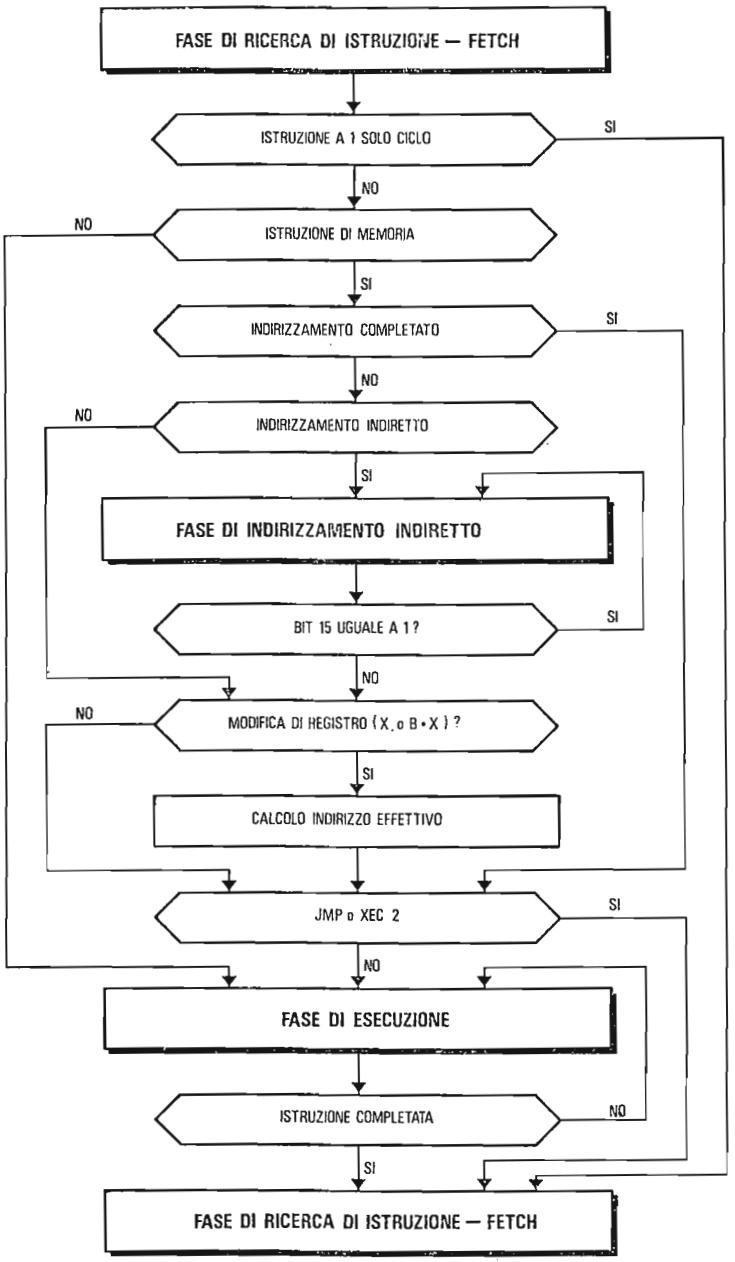

FASE DI RICERCA DI ISTRUZIONE — FETCH
ISTRUZIONE A 1 SOLO CICLO
SI
NO
NO
ISTRUZIONE DI MEMORIA
SI
INDIRIZZAMENTO COMPLETATO
SI
NO
NO
INDIRIZZAMENTO INDIRETTO
SI
FASE DI INDIRIZZAMENTO INDIRETTO
BIT 15 UGUALE A 1 ?
SI
NO
NO
MODIFICA DI REGISTRO (X, o B + X) ?
SI
CALCOLO INDIRIZZO EFFETTIVO
JMP o XEC 2
SI
NO
FASE DI ESECUZIONE
ISTRUZIONE COMPLETATA
NO
SI
FASE DI RICERCA DI ISTRUZIONE — FETCH

Fig. 2.11 -

secuzione (che variano a seconda del tipo di istruzione) possono, secondo i casi, essere con o senza ciclo di memoria. Alla fine di una fase di esecuzione si possono presentare due possibili casi:

- l'istruzione è stata completamente eseguita: viene impostata una fase di ricerca di istruzione;
- l'istruzione non è stata completamente eseguita: viene impostata una nuova fase di esecuzione.

Durante l'ultima fase di esecuzione (dell'eventuale catena di fasi di questo tipo) viene aggiornato - se è il caso - il contenuto del registro C (si pensi, ad esempio, ad una istruzione ISZ; se la condizione di salto condizionato è verificata, il contenuto del registro C deve essere ulteriormente incrementato di una unità); viene inoltre realizzato in ogni caso il trasferimento: (C)→L, di modo che l'indirizzo dell'istruzione successiva sia disponibile sia nel registro C che nel registro L.

Le considerazioni fatte possono essere riassunte nello schema di flusso della fig 2.11.

2.5. IL SISTEMA DI INGRESSO-USCITA.

In un minicomputer vi sono generalmente tre modi di comunicazione per l'ingresso e l'uscita dei dati fra le unità periferiche e l'unità centrale CPU:

- il modo su "controllo di programma" (programmed input-output);
- il modo su "interruzione automatica" (interrupt);
- il modo di "accesso diretto alla memoria" (DMA).

In molti minicomputers, qualunque sia il modo in esecuzione, il trasferimento dei dati avviene sempre attraverso il BUS-Ingresso-Uscita che è costituito da un sistema di linee parallele (dell'ordine di 50 ÷ 60 fili) su cui viaggiano i dati e i comandi per le unità di interfaccia.

La struttura delle linee risulta suddivisa nei seguenti gruppi:

- **Linee per la trasmissione dei dati e dei comandi (Data Bus-DB)** che costituisce un gruppo di 16 linee bidirezionali utilizzate per il trasferimento dei dati e dei comandi tra il registro accumulatore e le unità di interfaccia delle periferiche.

- **Linee per la trasmissione delle istruzioni (Device Address and Instruction Code Bus-DAB)** che costituisce un gruppo di 11 linee utilizzato per il trasferimento, dal registro d'istruzione I alle unità d'interfaccia delle periferiche, dei codici delle istruzioni d'ingresso-uscita. Con i fili da 0 a 5 trasmettono il codice d'identificazione della periferica e con i fili da 6 a 10 trasmettono la funzione richiesta, secondo quanto è specificato nei rispettivi campi dell'istruzione di ingresso-uscita.
- **Linee per la trasmissione degli indirizzi (Location Address Bus-LAB)** che costituisce un gruppo di 15 linee utilizzato per il trasferimento nel registro di indirizzo delle locazioni di arrivo delle richieste di interruzione automatica oppure della locazione di memoria con cui deve essere effettuato il trasferimento in accesso diretto (DMA).
- Linee di temporizzazione. Gruppo di 3 linee utilizzato per sincronizzare le periferiche con le fasi del controllo centrale.
- Linee di controllo. Gruppo di 5 linee utilizzate per il funzionamento in interruzione automatica ed in accesso diretto alla memoria (DMA).
- Catena di priorità. Viene utilizzata per gestire in modo gerarchico le richieste di interruzione automatica e di accesso diretto alla memoria (DMA).
- Linea di azzeramento iniziale. Viene utilizzata per azzerare tutti i registri delle periferiche.
- Linea di segnalazione di mancanza di rete. Viene utilizzata per segnalare sia la mancanza di rete, sia che la memoria è disabilitata.

Tutto questo sistema di linee parallele va collegato alle diverse unità di interfaccia. Ciascuna di queste unità può essere considerata come costituita da due parti: l'una, uguale per tutte le periferiche, costituisce la parte direttamente a contatto con l'insieme delle linee di ingresso-uscita, l'altra soddisfa le esigenze particolari per la gestione dei comandi e dei controlli della singola periferica.

In generale i circuiti caratteristici contenuti nell'unità d'interfaccia, ed indicati nello schema a blocchi della fig. 2.12, sono così suddivisi:

- un registro dati
- un registro di comando
- un registro di stato
- un dispositivo di riconoscimento del numero di identificazione della periferica

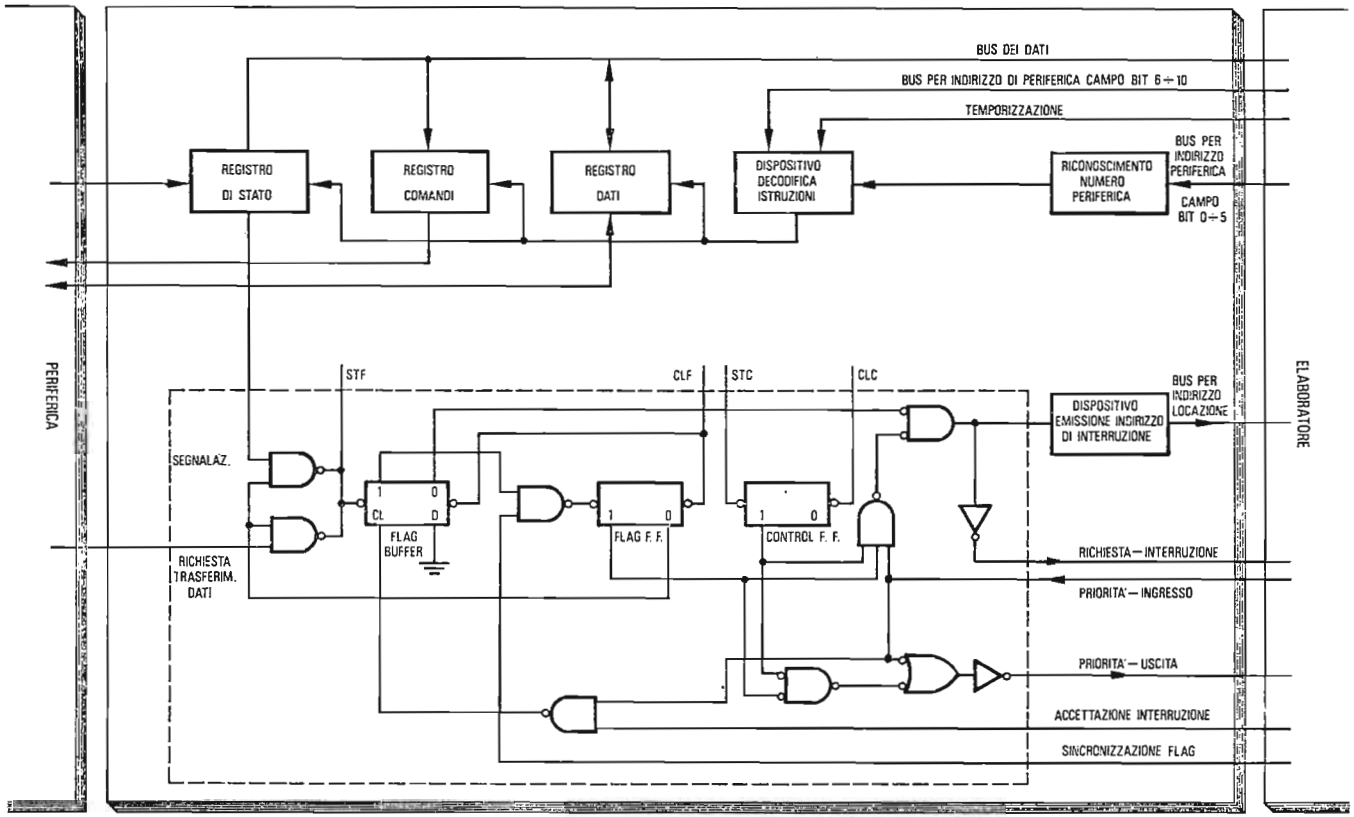


Fig. 2.12 - SCHEMA A BLOCCHI DI PRINCIPIO DI UNA UNITA' DI INTERFACCIA PER PERIFERICA CONNESSA AD UN MINICOMPUTER

- un dispositivo di decodifica delle istruzioni
- un bistabile detto "flag buffer" (FB) che memorizza, in modo asincrono rispetto all'unità centrale, sia le richieste di trasferimento dati, sia le segnalazioni da parte della periferica
- un bistabile, detto "flag" (F), mediante il quale le richieste precedentemente memorizzate nel "flag buffer" vengono sincronizzate con l'unità centrale
- un bistabile di controllo per l'abilitazione della periferica al funzionamento in interruzione automatica
- un bistabile per l'abilitazione della periferica al funzionamento in accesso diretto alla memoria
- un dispositivo per la formulazione delle richieste di interruzione automatica, di accesso diretto alla memoria e per la codifica delle medesime.

Si può così passare a descrivere le funzioni dei diversi modi di trasferimento dei dati sul sistema di ingresso-uscita così strutturato.

2.5.1. Modo su "Controllo di Programma".

La periferica è gestita in maniera sincrona con il programma il quale si occupa sia dell'operazione di sincronizzazione delle richieste della periferica con la disponibilità dell'elaboratore, sia del trasferimento dei dati.

Il programma provvede inoltre periodicamente a controllare lo stato della periferica e a gestire le condizioni anormali rilevate.

Ad esempio, prima che la CPU possa trasferire i dati alla periferica, essa deve controllare lo stato della periferica e determinare se è o no pronta ad accettare il trasferimento. Lo stato della periferica è controllato in generale dal bistabile "Flag" che quando è in 1 indica che la periferica è impegnata mentre quando è in 0 indica che i dati possono essere trasferiti.

2.5.2. Modo su "Interruzione Automatica" (Interrupt).

La maggioranza delle unità periferiche in genere possono trasferire i dati ad una velocità molto inferiore a quella con cui opera la CPU, sicché in un programma di trasferimento dati il tempo di lavoro della CPU sarebbe impiegato soprattutto nell'attesa che la periferica sia pronta ad accettare o trasmettere un dato.

L'utilizzazione della CPU può essere molto migliorata se essa continua a

svolgere altri programmi finché la periferica non si dichiari "pronta" per il colloquio. Questa possibilità è offerta dal modo di "interruzione automatica", disponibile su tutti i minicomputers, e che blocca il proseguimento del programma in corso nella CPU, il cui controllo viene invece ceduto ad un sottoprogramma di servizio detto appunto di "interruzione". Questo sottoprogramma individua quale periferica ha causato l'interruzione e trasferisce a sua volta il controllo al programma di servizio di questa stessa periferica.

Perché l'unità centrale possa correttamente riconoscere e soddisfare una richiesta di interruzione devono essere rispettate tre condizioni fondamentali:

a) la richiesta deve essere sincronizzata con l'orologio centrale del calcolatore,
b) richieste simultanee devono essere servite in sequenza,
c) l'interruzione del programma in corso deve sempre verificarsi, in modo che alla fine del sottoprogramma di servizio dell'interruzione sia possibile riprendere il programma originario.

Per soddisfare queste condizioni i segnali d'interruzione sono ordinati, **assegnando ad ogni segnale d'interruzione un livello in una gerarchia di priorità.**
Un segnale di qualsiasi livello può interrompere il programma principale, può interrompere le istruzioni di servizio relative ad un livello di priorità inferiore, non può interrompere le istruzioni di servizio relative ad un livello di priorità uguale o superiore.

Il sistema di interruzione, di cui la CPU è provvista, permette di sospendere l'esecuzione di un programma anche su richiesta proveniente dall'interno dell'unità di elaborazione centrale, oltre che da una periferica.

I segnali d'interruzione interni sono generati dal verificarsi di almeno una delle seguenti condizioni:

- Mancanza di alimentazione (Power failure)
- Violazione di memoria protetta e traboccamento di memoria (Memory protect and overflow)
- Errore di parità in memoria (Memory parity error).

Questi segnali occupano nell'ordine i primi 3 livelli della catena di priorità.

Le tecniche hardware e software per realizzare i modi di interruzione au-

tomatica variano da tipo a tipo di minicomputers e piuttosto che ad una trattazione generale è utile allora rimandare il lettore all'esame dettagliato del criterio adottato nel minicomputer che interessa.

2.5.3. Modo di accesso diretto alla memoria DMA.

Questo modo è molto importante tutte le volte che si vuole raggiungere la piú elevata velocità nel trasferimento dei dati fra la periferica e la memoria.

Il funzionamento in DMA viene anche detto in "cycle stealing mode": ciò deriva dal fatto che, quando una periferica può funzionare in DMA e ne fa richiesta, la CPU completa l'istruzione in corso e poi interrompe il programma in esecuzione in modo che per il ciclo successivo la periferica può trasferire un dato direttamente con la memoria.
Dopo che all'unità periferica è stato riservato questo ciclo (da cui la denominazione di "cycle stealing"), la CPU riprende a funzionare su controllo del programma. Il canale di DMA generalmente è realizzato con una unità di interfaccia che contiene i circuiti logici per selezionare l'indirizzo di memoria in cui accedere nel trasferimento del dato e per specificare il numero di parole da trasferire con la periferica posta sul canale DMA. In diversi calcolatori, come ad esempio il LABEN 70, il canale di DMA è realizzato con un unico circuito di controllo che può agire su una qualsiasi unità periferica collegata alla CPU anche attraverso il Bus di ingresso-uscita. In questa maniera non occorre creare canali e interfaccie speciali per il DMA, ma è sufficiente commutare in funzionamento DMA una delle periferiche già connesse sul canale di ingresso-uscita.

Riferendosi allo schema a blocchi del LABEN 70, si può osservare che per ottenere il funzionamento in DMA devono essere presenti nell'elaboratore tre sottosistemi: il comando di DMA da parte dell'unità di controllo, i circuiti logici propri del canale di DMA e la periferica provvista del dispositivo di chiamata in DMA sull'unità d'interfaccia. Il controllo di DMA è unico, mentre il numero di periferiche è vincolato solo ad essere minore di 64. E' sufficiente un unico canale per far lavorare in DMA qualsiasi di queste periferiche, purché una per volta. Il trasferimento viene governato da tre parametri precisati nei circuiti del canale di DMA: l'indirizzo d'inizio del trasferimento, contenuto in un contatore di indirizzo **Address Counter (AC)**; il numero (negativo) di dati da trasferire, contenuto in un contatore di parola **Word Counter (WC)**; il codice della periferica abilitata all'uso del canale di DMA e la modalità del trasferimento, contenuti in un **registro di controllo (CR)**.

In questo caso i circuiti dell'unità di controllo che presiedono sia all'esecuzione delle istruzioni, sia alla gestione del segnale di interruzione non intervengono nel trasferimento dei dati. Infatti esso è effettuato direttamente fra la periferica e la memoria, e l'esecuzione delle istruzioni viene interrotta finché la memoria viene usata dal sistema di DMA.
Il trasferimento in DMA si può applicare a blocchi di dati di dimensioni variabili da 1 a 32768 parole e può assumere diverse modalità: in ogni modalità il trasferimento di un dato occupa un solo ciclo di macchina.

Non si ha alcuna perdita di tempo nella commutazione fra programma in esecuzione e modo di funzionamento in DMA o viceversa. Rispetto ad un trasferimento compiuto a programma, sia pure in interruzione automatica e con le speciali istruzioni di trasferimento che richiedono sei cicli macchina per dato, si ha perciò un forte aumento di velocità di trasferimento ed anche una minore sottrazione di tempo di calcolo all'unità centrale.

Capitolo 3

SISTEMI DI MEMORIA.

3.1. GENERALITA'.

Precedentemente si è visto come possono essere collegati più flip-flop per formare un registro capace di immagazzinare un certo numero di bit. L'uso degli stessi per immagazzinare grandi quantità di dati si è rivelato nel passato così costoso da diventare proibitivo, e solo recentemente con l'avvento dei circuiti integrati a larga scala LSI e delle tecnologie MOS è stato possibile realizzare memorie di bistabili a semiconduttori a grande capacità di dati.

Nei sistemi di memoria il problema tecnico più impegnativo non è costituito tanto dagli elementi stessi dalla memoria, quanto dal complesso dei circuiti che permettono l'accesso alla memoria selezionando ogni specifico elemento per posizionarlo in uno stato o leggerne lo stato.

Esiste una grande varietà di sistemi e, mentre è necessario rimandare ai riferimenti bibliografici per quei tipi che sono rimasti allo stato di ricerca di laboratorio o che sono stati descritti solo recentemente in letteratura, esamineremo i seguenti che sono finora tra i più diffusi ed affermati nelle utilizzazioni:

1) **nuclei toroidali di materiale magnetico** come elementi di memoria e flussi concatenati di correnti in coincidenza come elementi di accesso;
2) **memorie a semiconduttore**, sia statiche che dinamiche, realizzate con tecnologia bipolare e MOS;

3) **memorie a trasferimento di carica (CCD)**, che vengono realizzate con materiale semiconduttore, ma con procedimenti di fabbricazione più semplici che non quelli impiegati nella tecnologia bipolare e nella tecnologia MOS;
4) **memorie a bolle magnetiche**, che sfruttano l'esistenza di domini magnetici cilindrici in certi materiali, per memorizzare l'informazione, con procedimenti di scrittura e lettura interamente elettronica;
5) **superfici di materiale magnetico**, in cui piccole areole magnetizzate costituiscono gli elementi di memoria alle quali si può accedere posizionandole di fronte a opportune testine di sonda.

3.2. LE TECNOLOGIE DELLE UNITA' DI MEMORIA.

La memoria è un sottosistema importante dell'elaboratore: da una memoria ideale si richiede una grande capacità, una velocità eguale a quella della logica di calcolo, un basso consumo, un piccolo ingombro, un basso costo. Tutti questi parametri non possono essere ottimizzati contemporaneamente e perciò i progettisti ricorrono a strutturare i sistemi di memoria con unità realizzate con tecnologie diverse e suddivise per funzioni specializzate. In sintesi tali strutture si possono schematizzare con una suddivisione delle unità su tre livelli:

- le unità di memoria temporanea (scratch-pads, buffer memorics), in generale costituite da registri per i dati in fase di elaborazione, che hanno la velocità più elevata ed una capacità limitata dal centinaio di bits ad alcune migliaia di bits in qualche caso;
- le memorie di lavoro, in passato costituite generalmente a nuclei di ferrite ed oggi realizzate anche con componenti a semiconduttori LSI, che memorizzano i programmi ed i dati in elaborazione e che devono essere perciò interrogate con continuità. I tempi di accesso sono nel campo del microsecondo e le capacità raggiungono anche i 10^8 bits;
- le memorie di archivio che devono immagazzinare l'insieme dei dati e programmi disponibili, nelle quali la velocità è sacrificata in funzione della capacità e del costo.

Nella Tabella 3.1 è riportato lo stato delle tecnologie usate nei diversi sistemi di memoria.

TABELLA 3.1 – TECNOLOGIE DEI SISTEMI DI MEMORIA

Tipo di tecnologia	Tempo di accesso	Capacità in bits	Costo per bit (Lit)
Memorie di lavoro			
Circuiti MOS-LSI	0.1 ÷ 1 μs	$10^4 \div 10^8$	0.1 ÷ 1
Circuiti bipolari LSI	50 ÷ 100 ns	$10^3 \div 10^6$	1 ÷ 10
Nuclei di ferrite	0.5 ÷ 10 μs	$10^4 \div 10^8$	10
CCD	100 μs	$10^5 \div 10^9$	0.01 ÷ 0.1
A bolle magnetiche	100 ÷ 1000 μs	$10^5 \div 10^9$	0.01 ÷ 0.1
Memorie di archivio			
Dischi magnetici	8 ÷ 200 ms	$10^6 \div 10^{12}$	0.01 ÷ 0.1
Nastri magnetici	0.1 ÷ 100 sec	10^6 ÷ ed oltre	0.0001 ÷ 0.01

Nella figura 3.1 si riporta l'andamento nel tempo del costo per byte in funzione del tempo di accesso per le unità di memoria con tecnologia diversa. Si osserva che esiste una vasta area che separa le memorie completamente elettroniche (a semiconduttore, a fili e a nuclei di ferrite), da quelle elettromeccaniche. Molte ricerche sono svolte per sviluppare unità che vadano a ricoprire tale area, perché ciò potrebbe portare a rivoluzionare i tradizionali rapporti gerarchici e condurre a strutture completamente nuove. La realizzazione di memorie seriali di massa basate su componenti a semiconduttori (tamburi LSI) o su nuovi materiali magnetici come le bolle magnetiche potrebbero costituire nel prossimo futuro una soluzione al problema.

E' opportuno notare che, nello sviluppo dei sistemi di memoria, sono importanti e determinanti per l'applicazione pratica non solo i problemi propri della tecnologia del supporto memorizzante, ma in egual misura i problemi di interfaccia elettrica e meccanica o ambientale. La circuiteria ne-

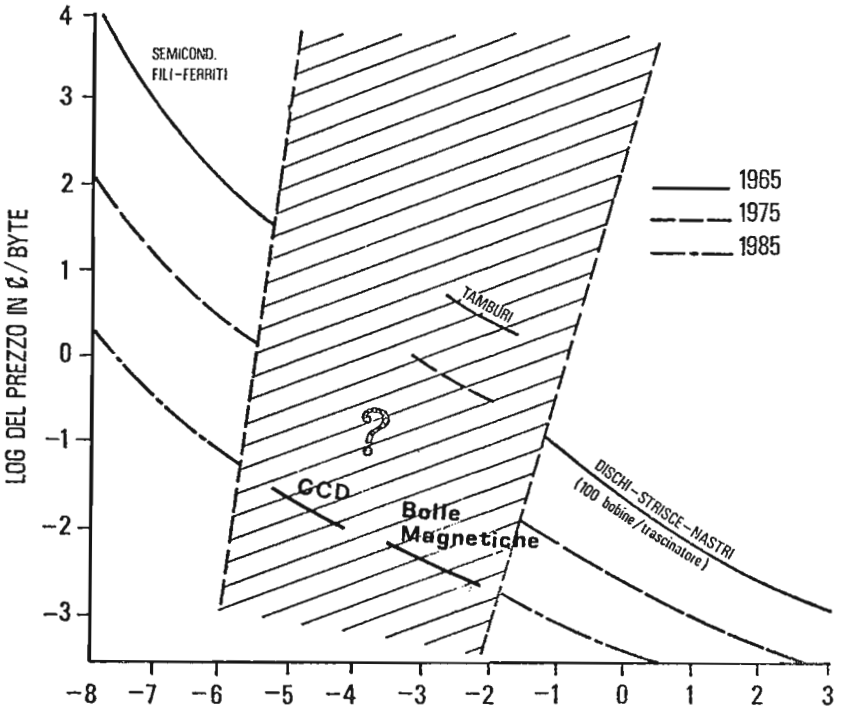


Fig. 3.1 -

cessaria per la selezione, la scrittura, la lettura, come pure l'esigenza di condizioni ambientali speciali, quali ad esempio temperatura o atmosfera particolarmente controllate, hanno spesso compromesso lo sviluppo industriale di nuove tecnologie inizialmente molto promettenti.

A questo proposito va notato che nelle memorie LSI a semiconduttore le celle sono identiche dal punto di vista fisico e tecnologico con i circuiti delle parti logiche del sistema. Scompare così il divario tra le rispettive velocità operative sicchè memoria e logica possono essere mescolate sulla stessa piastrina LSI.

Nelle memorie tradizionali a nuclei di ferrite si ottengono bassi costi solo realizzando piani di memoria di grandi dimensioni, in modo da diminuire l'incidenza percentuale del costo dei circuiti di selezione e comando. Le nuove memorie LSI non hanno più queste limitazioni in quanto, per i motivi anzidetti, l'andamento del costo per bit in funzione della capacità è circa uguale per memorie piccole e grandi. Questo fatto può modificare diverse concezioni nell'architettura dei sistemi di elaborazione dei dati.

Oggi è possibile suddividere la memoria di lavoro a semiconduttore in sottounità senza penalizzazioni di costo. Si attuano perciò sistemi con più sottounità in multiplex per migliorare la velocità, oppure si distribuiscono le unità di memoria associandole alle diverse unità operative in modo da renderle in ogni senso autonome per elaborazioni in parallelo.

3.3. STRUTTURE DELLE UNITA' DI MEMORIA.

Si può introdurre (passo di scrittura) o estrarre (passo di lettura) l'informazione in una data locazione di un sistema di memoria quando si dia lo indirizzo di detta locazione: si dice così che la memoria è indirizzabile per coordinate, cioè le locazioni di memoria sono accessibili specificandone le coordinate dell'indirizzo. Questo è in genere fornito in codice binario, sicché la lunghezza del codice - cioè il numero di bits contenuti nel codice - determina il massimo numero di locazioni indirizzabili. Ad esempio un indirizzo con 10 bits, può individuare $2^{10} = 1024$ locazioni, così come un codice con una lunghezza di 20 bits ne può individuare $2^{20} = 1.048.576$. Per ogni indirizzo si può memorizzare un dato avente tanti bits quanti sono gli elementi binari o celle elementari di memoria contenute nella locazione selezionata detta anche parola: il numero di bits memorizzabile in una locazione determina la cosiddetta lunghezza di parola.
La lunghezza di parola per la lunghezza di indirizzo determina il numero di celle memorizzabili e quindi **la capacità di memoria** in numero di bits. Si usa alle volte definire anche **la larghezza di banda** di un sistema di memoria: essa è data dal numero di bits immagazzinabili nell'unità di tempo, cioè dal rapporto tra il numero di bits immagazzinabili in un ciclo di memoria diviso per la durata del ciclo stesso. Così una memoria con parole da 16 bits e con ciclo di memoria di 1 microsecondo avrà una banda di 16 Milioni di bits al secondo.
Per quanto si riferisce alle unità di memoria a nuclei di ferrite, i sistemi di memoria sono formati a moduli componibili: un Modulo Base di Memoria (MBM) che contiene N parole, ciascuna con M bits, comprende in genere anche tutta la circuiteria per accettare un indirizzo, selezionare la locazione di parola corrispondente e scrivere o leggere il contenuto della parola.

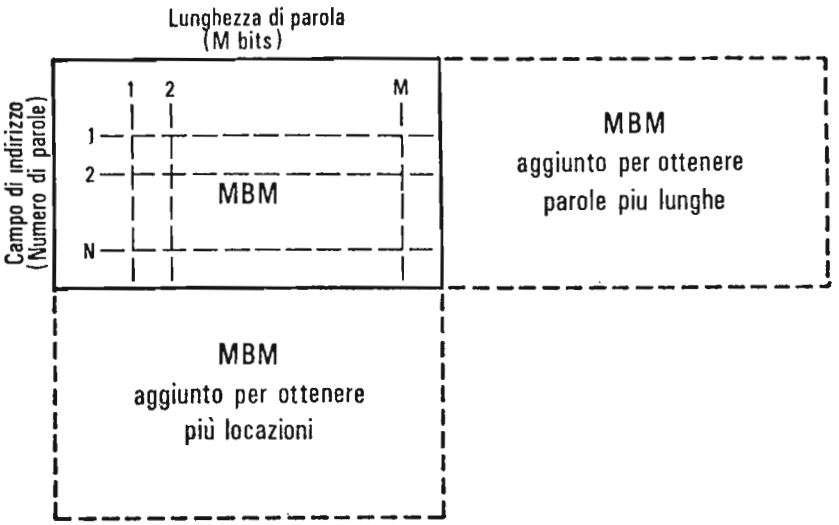


Fig. 3.2 - Il modulo base di memoria (MBM) mostrato entro il rettangolo a linee piene può essere espanso con l'aggiunta di altri moduli per aumentare sia la lunghezza di parola sia la lunghezza di indirizzo.

Ponendo più moduli uno accanto all.altro, come nello schema di figura, si può aumentare la lunghezza di parola, mentre ponendoli uno sopra all'altro si aumenta il numero di parole.

Normalmente nei minicomputers l'espansione di memoria avviene lasciando inalterata la lunghezza di parola ed aumentando il numero di parole: ad esempio si procede per blocchi da 4096 parole (4 K) di 16 bits o di 8 bits.

Quando l'espansione avviene aumentando la lunghezza di parola, oltre alla capacità di memoria aumenta anche la banda. Ma se la struttura logica e circuitale del calcolatore non è in grado di utilizzare una parola più lunga, la banda può essere aumentata adottando un comando a divisione di tempo per i diversi moduli di memoria posti uno accanto all'altro: questo metodo detto di "interfoglio" (interleaving) permette di andare avanti nei passi successivi del ciclo di memoria del primo MBM (selezione, lettura, scrittura), mentre vengono selezionati con sequenza temporale sfasata di un passo il secondo modulo, il terzo e così via.

Nella memoria si possono anche compiere operazioni di controllo sull'informazione immagazzinata. Ad esempio memorizzando nella parola un bit di parità con ogni dato, si ottiene la possibilità di controllare se il nu-

mero dei bits cambia o rimane pari o dispari: è quanto viene indicato come "**controllo di parità**". Altre forme più sofisticate di codificazione sono in grado di mettere in evidenza e correggere errori multipli nella parola.

L'informazione nella memoria può anche essere salvaguardata da un uso o da modifiche non consentite o non autorizzate: ad esempio programmi già collaudati residenti in memoria devono essere protetti da alterazioni generate da errori in programmi nuovi o da inserimenti sbagliati di programmi non autorizzati. I meccanismi di protezione possono allora consistere di bits addizionali nella parola il cui codice costituisce una specie di chiave per impedire cambiamenti non voluti.

Fra le diverse funzioni che si possono affidare alla memoria non è esclusa nemmeno quella di effettuare operazioni di calcolo: ad esempio si può effettuare nella stessa memoria la moltiplicazione se si memorizza una tavola pitagorica e si individua il prodotto nella locazione selezionata con gli indirizzi del moltiplicando e del moltiplicatore.

Come si è visto, i sistemi di memoria possono essere classificati e suddivisi in diverse categorie a seconda del tipo di tecnologia adottato e del modo di funzionamento consentito. Spesso però la classificazione non è univoca perché si possono avere sovrapposizioni di caratteristiche e prestazioni fra tipi diversi.

3.4. MEMORIE STATICHE E DINAMICHE.

Una prima classificazione divide le memorie in **statiche**, quando l'indirizzamento è fatto con mezzi elettronici oppure ottici, ed in **dinamiche** quando per individuare un indirizzo occorre muovere meccanicamente il supporto di memoria (disco o nastro magnetico) rispetto alle sonde di lettura e scrittura.

Il tempo di accesso nelle memorie dinamiche non è costante ed indipendente dall'indirizzo: ad esempio se in un disco rotante a teste fisse si considerano due indirizzi adiacenti di cui uno immediatamente sotto la testina di lettura e l'altro appena oltrepassato, il tempo di accesso per il primo è istantaneo mentre per il secondo occorre attendere che il disco compia un giro. Per questo motivo, ed anche per la inerente maggiore lentezza dei sistemi meccanici rispetto a quelli elettronici, le memorie statiche tendono

ad essere usate come memorie di lavoro nell'elaboratore centrale CPU, mentre quelle dinamiche sono generalmente impiegate come memorie di massa o di archivio.

3.5. MEMORIE STATICHE AD ACCESSO CASUALE: RAM (Random Access Memory).

Le memorie RAM ad accesso casuale, dette spesso anche memorie ad accesso diretto, hanno un tempo di lettura e scrittura che è sempre lo stesso qualsiasi sia l'indirizzo da selezionare: cioè in queste memorie l'informazione può essere scritta o letta selezionando e accedendo all'istante a qualsiasi indirizzo desiderato.

Le memorie a nuclei di ferrite, le più recenti memorie con transistori bipolari ℓ a MOS a circuiti integrati, sono organizzate nella quasi totalità dei casi per funzionare come memorie RAM. La loro applicazione s'impone soprattutto dove è necessario raggiungere massima velocità di accesso e flessibilità di struttura modulare: cioè come memorie di lavoro nelle unità centrali dei calcolatori.

3.6. MEMORIE STATICHE SEQUENZIALI.

Questo tipo di memoria coincideva nel passato con le memorie a linea di ritardo, in cui l'informazione viene mantenuta facendola ricircolare ininterrottamente tra entrata ed uscita. Esse accettano all'ingresso i dati ad una certa frequenza f e li resistuiscono in uscita dopo un tempo ΔT pari al ritardo della linea: la capacità di memoria è data ovviamente dal prodotto $(f \cdot \Delta T)$.

Oggi in questo tipo di memoria rientrano anche le memorie a transistori MOS organizzate come registri a scorrimento (shift registers).

Le memorie del tipo "registro a scorrimento" differiscono dalle memorie a linea in quanto ciò che è fisso è il numero di celle di memorie o di locazione, mentre il tempo Δt di propagazione fra ingresso ed uscita può variare in funzione della frequenza degli impulsi con cui si comanda il ritmo di scorrimento nel registro.

Il tempo di accesso è ovviamente funzione del tempo di transito fra entra-

ta ed uscita e può essere ridotto organizzando la memoria in modo da avere, per una stessa capacità, più registri di scorrimento più corti in parallelo invece di un unico registro più lungo.

Le memorie sequenziali MOS trovano la loro applicazione più conveniente nelle unità di visualizzazione dei dati (display terminals), nei modem, nelle unità di concentrazione e trasmissione di dati, nei filtri digitali, ed infine come memoria di massa quando la loro capacità si estende a valori di 10^6 bits ed oltre: in quest'ultimo caso esse simulano praticamente i dischi ed i tamburi magnetici, rispetto ai quali presentano non solo vantaggi di velocità e di costo ma anche di durata e di affidabilità.

3.7. MEMORIE ROM (Read-Only-Memories).

Le memorie ROM, come indica il nome, sono memorie adatte per la sola lettura: praticamente se si considera una memoria RAM e le si toglie la capacità di scrittura, si ottiene una ROM. Esse perciò sono memorie ad accesso casuale o accesso diretto, ma molto più economiche delle RAM proprio a causa della eliminazione della capacità di scrittura.

Le ROM possono diversificarsi a seconda del modo con cui è possibile inserire e scrivere inizialmente l'informazione che poi deve essere letta. I diversi tipi possono essere classificati come segue:

- ROM inalterabili dopo il processo di costruzione
- PROM (Programmable ROM) il cui contenuto è alterabile con metodi elettronici, ma non durante il funzionamento della macchina
- EPROM il cui contenuto è non solo programmabile ma anche cancellabile più volte e quindi riprogrammabile
- RMM (Read Mostly Memories) in cui il tempo di scrittura è molto più lungo di quello di lettura.

Le ROM inalterabili sono quelle che vengono prodotte dal costruttore scrivendo direttamente durante il processo di fabbricazione la configurazione di 0 e 1 richiesta dall'utilizzatore. Nel caso di memorie a semiconduttore si realizzano maschere per la deposizione o diffusione del circuito integrato "custom tailored", cioè fatte apposta per il cliente. Successivamente non vi è alcun mezzo per modificarne il contenuto.

Nelle memorie PROM l'informazione da memorizzare può essere riscritta e

modificata, ma il tempo necessario per la scrittura risulta molto più lungo del tempo di lettura ed i metodi di scrittura sono completamente diversi dal metodo elettronico di lettura. Ad esempio la cancellazione e la riscrittura di nuovi dati nelle celle di memoria a semiconduttore può avvenire mediante l'uso di una maschera ottica illuminata da una sorgente di luce ultravioletta; oppure si può modificare il contenuto di una cella inviando ai terminali opportunamente predisposti una corrente predeterminata che modifica i cammini di corrente e le connessioni nell'interno della cella. In questi casi il contenuto della ROM può essere modificato, ma solo in laboratorio dal progettista del sistema elettronico in cui la memoria va inserita e perciò essa viene detta "programmabile".

Vi sono infine le memorie RMM (Read-Mostly-Memories) che sono ROM in cui è possibile riscrivere l'informazione senza estrarle dal sistema in cui sono inserite e senza sottoporle perciò a processi speciali in laboratorio. In esse il tempo di scrittura è abbastanza lungo, dell'ordine dei millisecondi, rispetto ai tempi di scrittura che possono andare dal microsecondo al nanosecondo. Generalmente richiedono impulsi di scrittura con ampiezza di tensione, intensità di corrente e durata che sono di alcuni ordini di grandezza superiori agli impulsi necessari per la lettura.

Lo sviluppo e l'uso delle memorie ROM si è diffuso soprattutto con l'applicazione delle tecniche di microprogrammazione. Si tende infatti a sostituire i circuiti logici sequenziali con memorie ROM immagazzinando in esse le istruzioni corrispondenti agli stati del sistema sequenziale che comanda lo svolgimento di un programma. Sicché tutti i microprogrammi relativi a procedure standard, gli insiemi di dati tabulati che devono essere a disposizione per riferimento frequente, come pure le sequenze dell'unità di controllo del calcolatore, vengono oggi attuati con memorizzazione in memorie ROM: ciò rende molto più economica, più affidabile e più flessibile la stessa struttura dell'unità centrale del calcolatore.

Spesso i microprogrammi ed i tabulati anzidetti vengono sviluppati e controllati prima su memorie RAM in cui è facile introdurre modifiche e fare adattamenti; ma quando la versione finale è stata collaudata, si preferisce immagazzinarli successivamente in memorie ROM ad alta velocità di lettura, non solo per motivi di economia e di velocità ma anche perché si è garantiti da qualsiasi incidente che viceversa nella RAM potrebbe alterare i dati memorizzati.

3.8. MEMORIE A NUCLEI MAGNETICI

In questo tipo di memoria vengono usati nuclei magnetici di ferrite a forma toroidale, caratterizzati da un ciclo di isteresi quasi rettangolare e da un elevato valore del flusso residuo.

Ogni nucleo può assumere due distinti stati di magnetizzazione residua: $+B_1$ (stato 1) e $-B_1$ (stato 0) indicati in fig. 3.3 (a). Esso va quindi considerato come **elemento bistabile capace di immagazzinare un bit.**

Per commutare lo stato da $-B_1$ a $+B_1$ occorre inviare attraverso il nucleo un impulso di corrente di ampiezza maggiore o eguale a ΔI_m come in fig. 3.3 (b). Infatti se l'impulso di corrente inviato fosse minore di ΔI_m, si avrebbe un piccolo ciclo di isteresi senza commutazione, ma con una piccola demagnetizzazione e gli impulsi alla commutazione risultano più piccoli. Invece inviando un impulso maggiore di ΔI_m, durante l'impulso stesso il nucleo passa da $-B_1$ al valore di saturazione $+B_m$ ed alla fine dell'impulso ΔI_m, il nucleo si posiziona al valore $+B_1$. Se l'impulso di corrente della stessa ampiezza viene inviato in senso opposto il nucleo passa da $+B_1$ a $-B_1$.

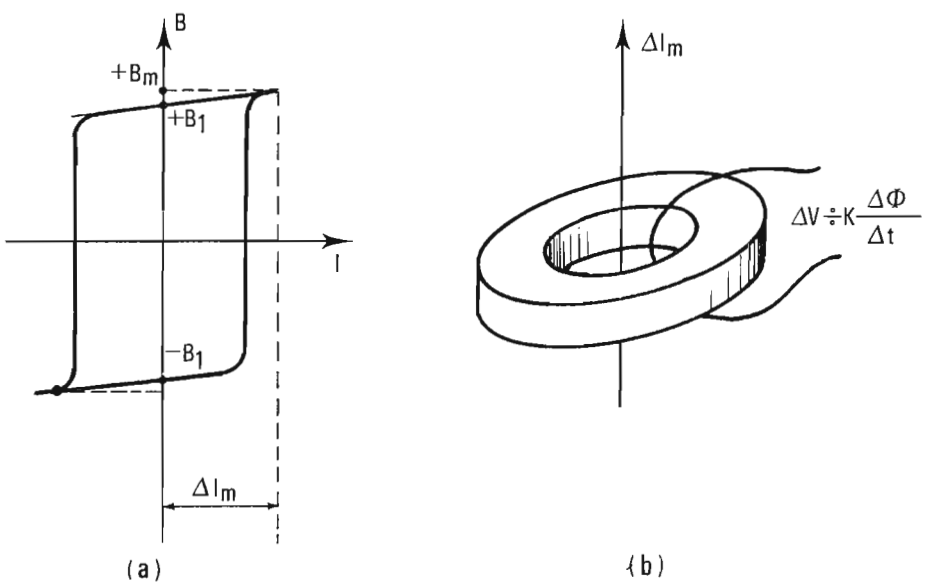


Fig. 3.3 - (a) Ciclo di isteresi di un nucleo di ferrite - (b) Impulso di comando ΔI_m e segnale ΔV indotto dalla commutazione del nucleo.

Durante la commutazione dello stato di magnetizzazione del nucleo si ha una forza elettromotrice indotta proporzionale alla velocità di cambiamento del flusso $\Delta \emptyset / \Delta t$, che può essere rivelata con un filo concatenato al nucleo: questa proprietà viene utilizzata per rivelare lo stato del nucleo, durante le commutazioni, attraverso il filo di sonda (sense wire).

Rispetto ai bistabili a transistori, **il nucleo di ferrite ha il vantaggio di non consumare potenza nei suoi stati stazionari e di mantenere il proprio stato anche quando viene a mancare la tensione di alimentazione ai circuiti (*).**

Nel progetto di ogni memoria a nuclei di ferrite, il problema principale è posto dal sistema di accesso per selezionare il nucleo di cui si vuole commutare lo stato onde scrivere o leggere l'informazione binaria.

Qualsiasi sia il metodo di selezione, **il nucleo viene sempre attraversato da almeno due o più fili attraverso i quali si inviano impulsi di corrente coincidenti nel tempo.**

Ad esempio nel caso di due soli fili, se inviamo una corrente $\Delta I_m/2$ in ogni filo, oppure una corrente ΔI_m in un solo filo, abbiamo o meno la commutazione del nucleo secondo quanto è illustrato nella Tabella 3.2 in funzione della somma totale delle correnti e dello stato precedente del nucleo.

TABELLA 3.2

Stato originale del nucleo	Impulso di corrente	Stato finale del nucleo	Commutazione di stato
0	$+ I_m$	1	si
0	$+ I_m/2$	0	no
0	$- I_m/2$	0	no
0	$- I_m$	0	no
1	$+ I_m$	1	no
1	$+ I_m/2$	1	no
1	$- I_m/2$	1	no
1	$- I_m$	0	si

(*) Una memoria capace di mantenere inalterate le informazioni immagazzinate, anche in assenza di tensione di alimentazione, viene detta non-volatile. Hanno questa proprietà tutte le memorie di tipo magnetico a nuclei, a nastro, a tamburo.

3.8.1. Metodo di selezione 2D.

Nella sua forma più semplice tutta la memoria può essere considerata disposta su un piano (2 dimensioni) e costituita da una matrice a nuclei di ferrite che può essere selezionata con 2^n fili di "parola" e con m fili di "digit", come indicato in fig. 3.4. Quando la memoria è in funzione, il ciclo di commutazione è composto di due passi: "lettura" e "scrittura". Nel **passo di lettura** (vedi fig. 3.5) si manda un impulso di corrente ΔI_m sulla linea di parola selezionata, cosicchè tutti i nuclei di questa linea sono portati allo stesso stato di magnetizzazione. I nuclei che erano prima nello

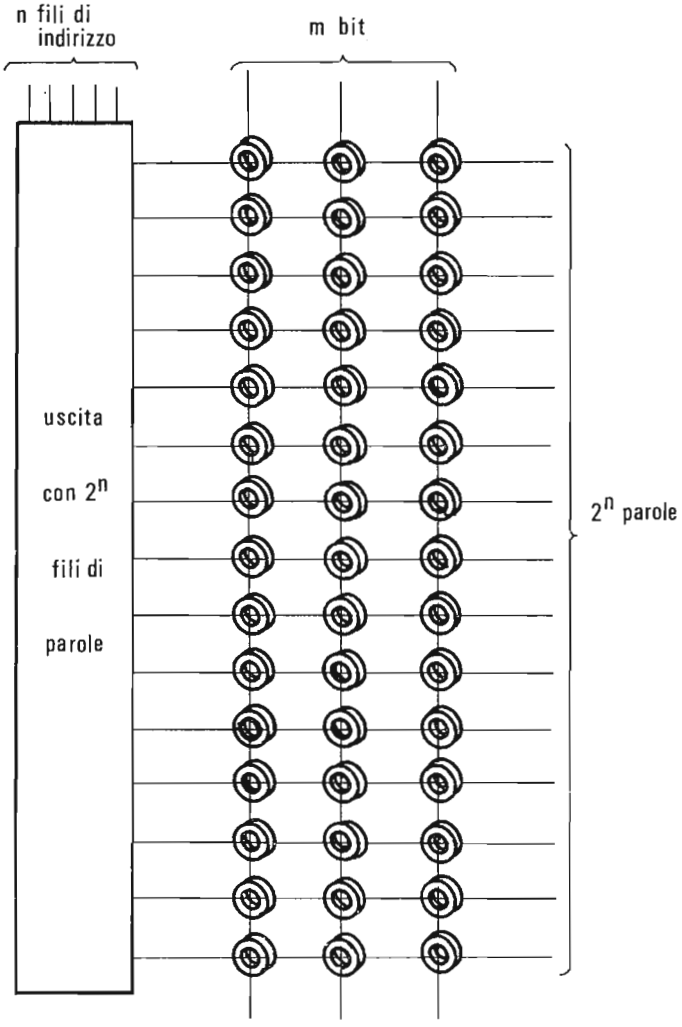


Fig. 3.4 - Selezione 2D, con 2^n parole ciascuna di m bit, cosicchè la capacità di memoria è m. 2^n bit. Nell'esempio n è uguale a 4 che dà luogo attraverso la matrice di decodifica a $2^4 = 16$ fili di selezione parole, ed n è uguale a 3 file di selezione di bit, cosicchè la capacità totale è di 48 bit.

stato opposto vengono commutati e ciò induce un impulso di tensione sui corrispondenti fili di bit, i quali così agiscono come fili di sonda e rivelano lo stato del nucleo. E' bene osservare che questa rivelazione viene fatta **in modo distruttivo**, in modo cioè che alla fine dell'impulso di sonda tutti i nuclei della fila di parola sono nello stesso stato e perciò l'informazione prima contenuta nei nuclei è andata perduta.
Nel **passo di scrittura** successivo l'impulso di corrente nella linea di parola cambia segno e viene ridotto in ampiezza a $-\Delta I_m/2$. Simultaneamente viene però inviato un altro impulso di corrente $\Delta I_m/2$ solo in quei fili di bit i cui nuclei devono essere posizionati nello stato 1: sicchè alla fine del passo di scrittura saranno riportati in 1 solo quei nuclei che hanno ricevuto in coincidenza l'impulso $\Delta I_m/2$ di parola e l'impulso $\Delta I_m/2$ di bit. Fino a che l'informazione non viene riscritta essa si trova perciò su un circuito di registro esterno piuttosto che nei nuclei di ferrite. In alternativa, si può riscrivere in memoria una informazione diversa da quella che si era rivelata nel passo di lettura.

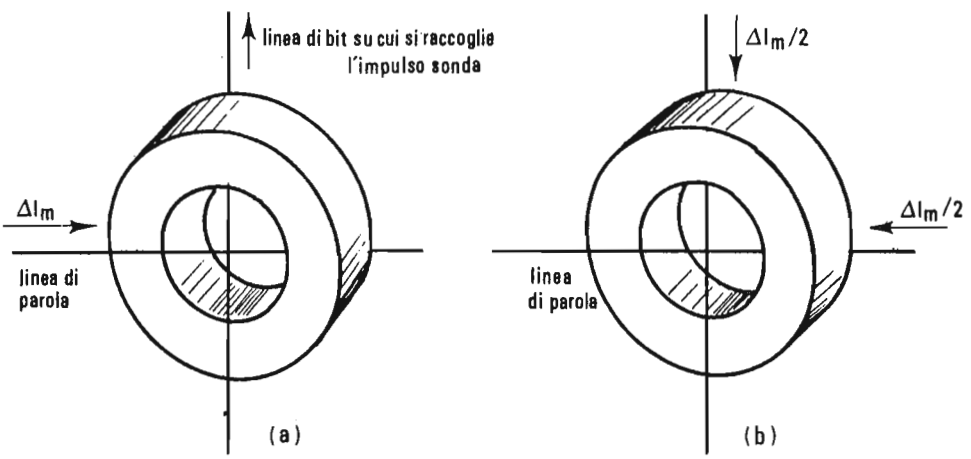


Fig. 3.5 - (a) Passo di lettura - (b) Passo di scrittura.

Nella configurazione 2D la memoria si dice anche organizzata "a parole". E' evidente che **ogni linea di parola deve essere comandata con un componente attivo, diodo o transistore, per essere selezionata**: cioè una memoria

a 16 K (16.384 parole) richiederebbe così 16.384 componenti per le altrettante linee di parola. Sicchè, quando la memoria ha una elevata capacità, questo tipo di selezione non è pratico e non viene seguito a causa dell'elevato numero di componenti richiesto per la selezione.

3.8.2. Metodo di selezione 3D.

Il concetto delle correnti coincidenti rende possibile un altro metodo di selezione detto 3D, che richiede un numero più limitato di componenti per la selezione.
Ogni nucleo è attraversato in questo caso da 4 fili: due fili di selezione X ed Y, un filo di "sonda" (sense), ed un filo di "inibizione" (inhibit).
Come indicato in fig. 3.6 la memoria è organizzata con tanti piani quanti sono i bit di parola. Ogni piano ha tanti nuclei quante sono le parole ed i nuclei sono dislocati a matrice quadrata o rettangolare.
Ad esempio una memoria di 16 K parole con 32 bits per parola avrà 32 piani ognuno con 16.384 nuclei (2^{14}) distribuiti a matrice quadrata (128 x 128) o rettangolare.
In questa maniera si vede che per selezionare 16.384 parole bastano 128 + 128 fili, oppure 256 + 64 fili e così via.
Con l'aiuto della fig. 3.7 si può illustrare come avviene l'operazione di lettura e di scrittura.
Sui fili verticali X_i e orizzontali Y_i vengono inviati in coincidenza impulsi di corrente con ampiezza $\Delta I_m/2$. Si provoca così la commutazione del nucleo individuato dall'incrocio dei due fili prescelti, mentre tutti gli altri nuclei della stessa riga o della stessa colonna non possono commutare essendo concatenati solo con una corrente $\Delta I_m/2$.
Durante il **"passo di lettura"** il verso degli impulsi di corrente sui fili X_i e Y_i è tale che **se il nucleo era nello stato zero esso vi rimane, mentre se era nello stato uno viene commutato.** Il cambiamento di flusso che ne risulta induce un impulso di tensione nel filo di sondaS dato che questo attraversa tutti i nuclei del piano. Questo metodo di lettura è anch'essa distruttivo, sicchè deve essere completato dal **"passo di scrittura".**
In questo secondo passo la corrente viene inviata sui fili X_i ed Y_i in senso opposto a quello usato durante la lettura. **L'impulso di selezione che ne risulta deve tuttavia commutare nello stato 1 solo i nuclei di quei piani che**

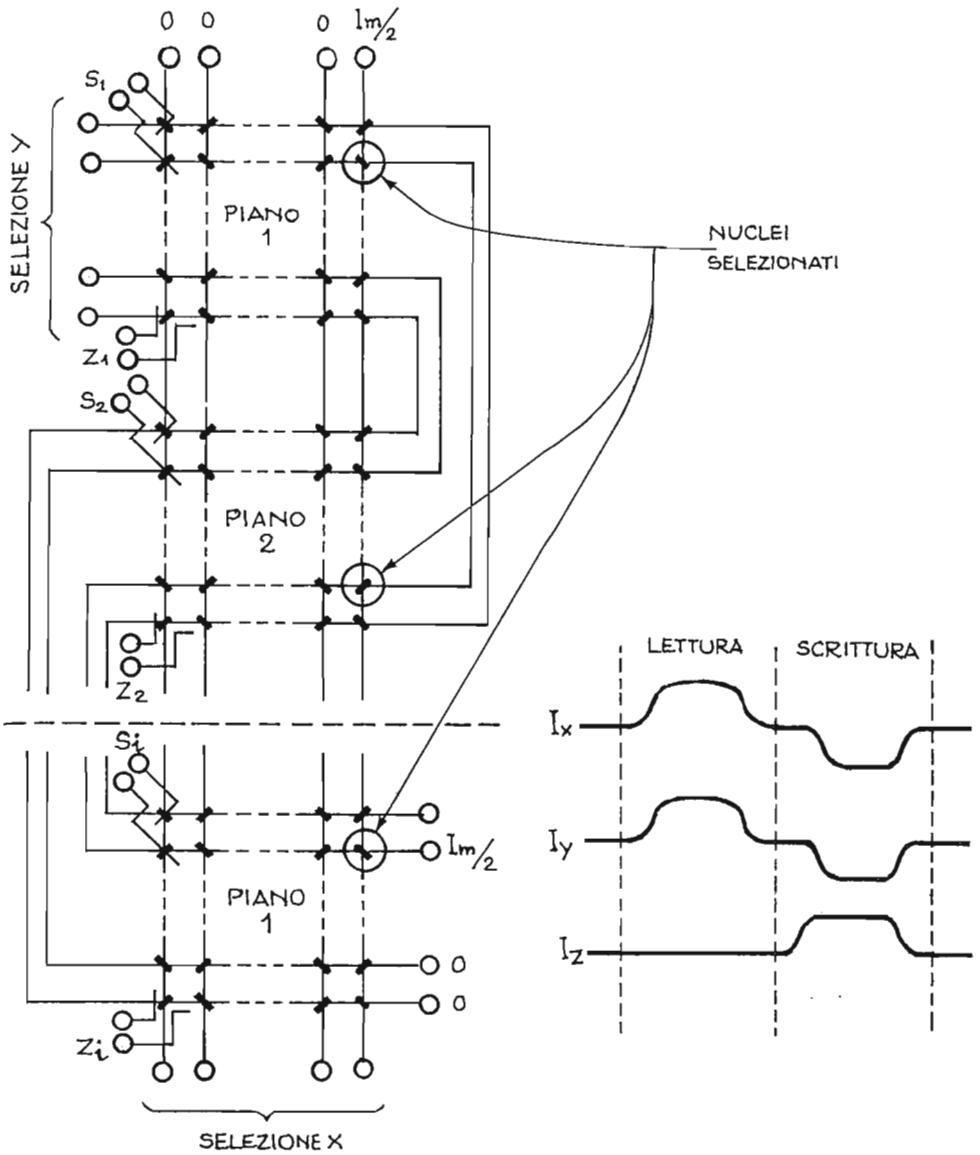


Fig. 3.6 — Memoria organizzata con la selezione 3D: a) selezione dei nuclei. I fili di selezioni X ed Y sono collegate in serie con tutti i piani; due altri fili di "sonda" (S) e di "inibizione" (Z) sono invece inseriti separatamente su tutti i nuclei di ogni piano; b) forme d'onda di selezione.

corrispondono ai bit con valore 1 della parola che si sta scrivendo. Nei piani i cui bit devono rimanere in 0 si invia perciò un impulso di corrente $\Delta I_m/2$ nel filo di inibizione che presentandosi in coincidenza ed in senso opposto alla corrente di selezione dei fili Y_i, mantiene la corrente totale concatenata col nucleo al di sotto del valore ΔI_m necessario per la commutazione.

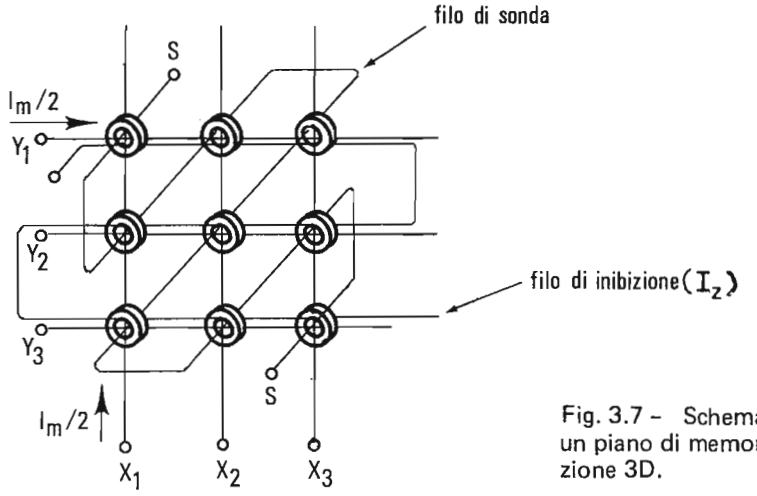


Fig. 3.7 - Schema di selezione per un piano di memoria con organizzazione 3D.

Per operare con il metodo delle correnti coincidenti è necessario che tutti i nuclei abbiano caratteristiche magnetiche note ed uniformi in modo che i valori di saturazione e di flusso residuo siano identici o quasi per tutti i nuclei.

Un limite alla capacità delle memorie con selezione 3D è posto dagli impulsi spuri forniti anche dai nuclei che ricevono solo la corrente $\Delta I_m/2$. Infatti il ciclo di isteresi non è perfettamente rettangolare sicchè l'impulso $\Delta I_m/2$ provoca nel nucleo semiselezionato un piccolo ciclo di isteresi con un ritorno ad un valore di magnetizzazione residua leggermente inferiore a B_1. Questi numerosi piccoli impulsi di tutti i nuclei semiselezionati, possono essere in parte cancellati con una opportuna scelta del cammino del filo di sonda: **si può fare in modo che il filo di sonda incontri i nuclei di una stessa riga o colonna in senso alterno in modo che l'impulso indotto dal nucleo delle file pari abbia segno opposto a quello delle file dispari.** Tuttavia questa compensazione non è mai perfetta e perciò si incontra alla fine un limite pratico oltre il quale il funzionamento diventa critico.

3.8.3. Metodo di selezione $2^1/_2$ D.

Questo metodo è intermedio fra i due precedentemente descritti. Esso richiede più componenti esterni per la selezione che il metodo 3D, ma ancora meno del metodo 2D. Per altro, rispetto al metodo 3D **riduce le capacità parassite e le induttanze dei fili di selezione X e rende perciò la memoria più rapida anche a grandi capacità.**

L'organizzazione $2^1/_2$ D è schematizzata in fig. 3.8

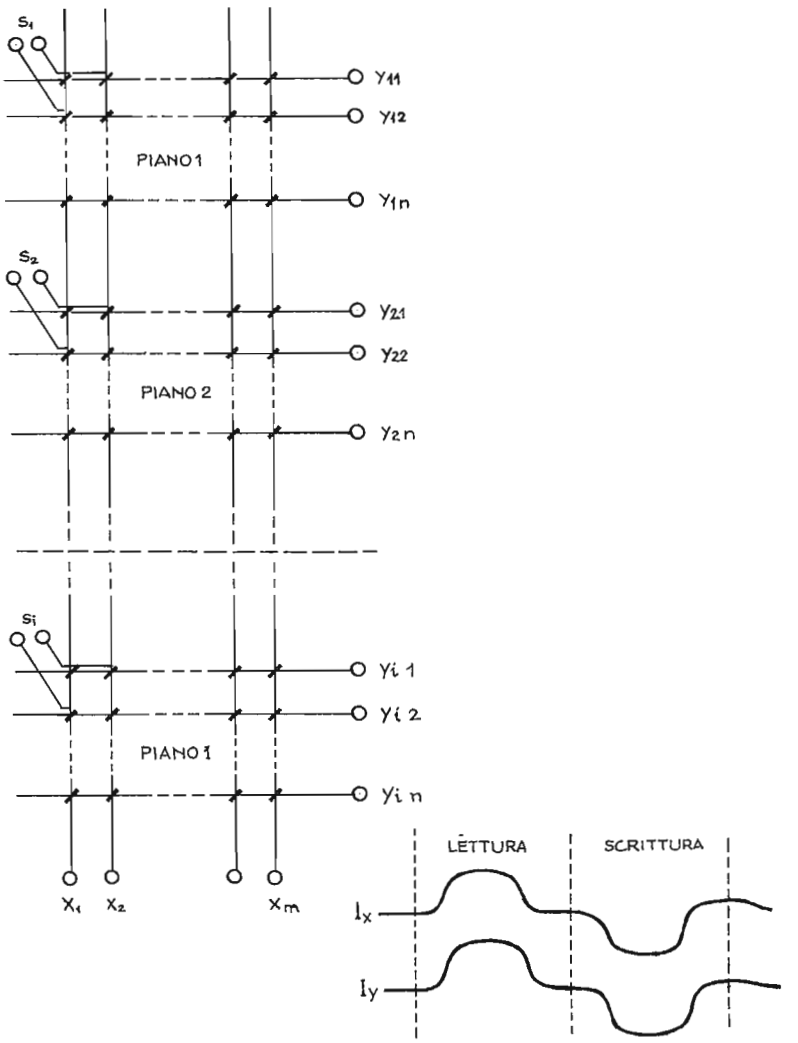


Fig. 3.8 — Schema di selezione $2^1/_2$ D: a) organizzazione dei nuclei, b) forme d'onda di selezione.

Il "**passo di lettura**" è compiuto nella stessa maniera che nel metodo 3D. Viceversa durante il "**passo di scrittura**" il filo di inibizione viene rimpiazzato introducendo tanti gruppi di linee di selezione Y quanti sono i piani: cioè i piani non ricevono tutti gli stessi fili Y, ma ogni piano ha il suo gruppo indipendente.
Il gruppo di linee Y in ciascun piano viene quindi eccitato oppure no dal comando "digit" a seconda che si debba scrivere 1 oppure 0. Sicchè se si deve scrivere 1, si avranno gli impulsi di corrente $\Delta I_m/2$ sul filo X e sul filo Y, mentre se si deve scrivere 0, il segnale Y mancherà perchè il comando "digit" non chiude la corrispondente linea di interruttori.

3.9. LE MEMORIE A SEMICONDUTTORE.

3.9.1. Generalità.

Accanto al criterio di memorizzazione a bistabili sono nati altri criteri di memorizzazione dell'informazione basati sulle proprietà peculiari dei MOS, di modo che attualmente si può parlare di due classi di soluzioni circuitali per le memorie a semiconduttore:

- le soluzioni basate sull'impiego di bistabili realizzati sia con tecnologia MOS che con tecnologia bipolare, che costituiscono l'insieme delle memorie statiche a semiconduttore;
- le soluzioni basate sull'impiego di meccanismi peculiari dei MOS con carica e scarica di condensatori, che costituiscono l'insieme delle memorie dinamiche a semiconduttore.

E' bene comunque precisare che qualunque sia il tipo di soluzione adottata per la memorizzazione a semiconduttori, la struttura logica delle memorie RAM e ROM rimane sempre quella descritta al par. 3.5 e 3.7.

Dal punto di vista tecnologico si registra una convergenza massiccia verso il MOS a canale N, che consente di ottenere, tanto nelle memorie statiche quanto in quelle dinamiche, migliori risultati dal punto di vista della velocità e della superficie di silicio occupata, grazie alla maggiore mobilità delle cariche.

3.9.2. Le memorie RAM a semiconduttore.

RAM bipolari statiche.

Indipendentemente dalla tecnologia utilizzata, la cella elementare di memoria è il bistabile, ottenuto accoppiando simmetricamente due circuiti invertitori costituiti ciascuno da un transistore e da una resistenza di carico.

La cella elementare di memorizzazione è realizzata con tecnologia a diffusione di collettore o con tecnologia Isoplanare, (fig. 3.9), gli emettitori E' ed E" dei due transistori multiemettitori sono connessi ambedue alla stessa riga di parola, mentre l'emettitore E_1 è collegato alla linea di colonna (o linea di bit) e l'emettitore E_2 è collegato alla linea di colonna complementare (o linea di $\overline{bit}$ (bit negato)). Le linee di colonna vengono utilizzate per selezionare la cella e per inviare i dati letti o scritti.

Facendo variare il livello della linea di parola si esegue l'operazione di **lettura** ed il dato compare sugli emettitori E_1 ed E_2 in modo da poter essere rivelato dall'amplificatore connesso alle linee di bit.

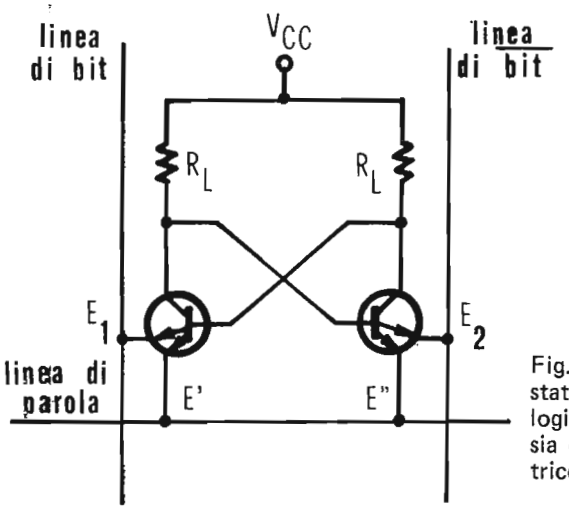


Fig. 3.9 – Cella elementare di memoria statica bipolare realizzabile sia con tecnologia Collector Diffusion Isolation (CDI) sia con tecnologia ad Isolamento Dielettrico (Isoplanare).

Analoga all'operazione di lettura è l'operazione di scrittura con l'unica differenza che in questo caso lo stesso dato ed il suo complemento vengono inviati sulle linee di bit.

Il funzionamento della cella elementare di memoria realizzata con tecnologia Schottky, (fig. 3.10), è analogo a quello descritto nel caso della tecnologia CDI ed Isoplanare. Gli emettitori E' ed E" dei due transistori sono

connessi entrambi alla stessa riga di parola, mentre i due collettori C' e C'' sono connessi tramite diodo Schottky alle linee di colonna (o di bit) e di colonna negata (o di $\overline{bit}$).

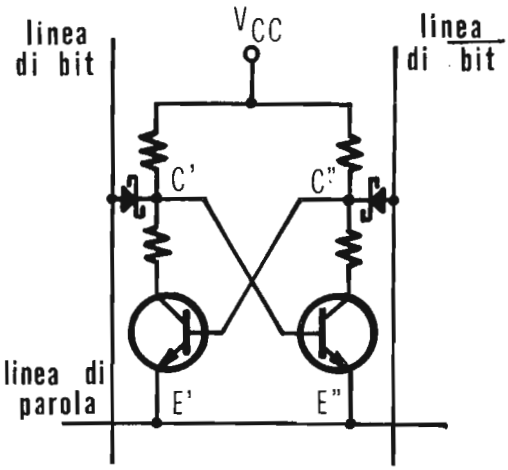


Fig. 3.10 — Cella elementare di memoria statica bipolare realizzabile con diodo Schottky.

Per effettuare l'operazione di lettura si varia il livello della riga di parola in modo che il dato compaia sui collettori C' e C'' per essere poi rivelato dall'amplificatore delle linee di bit. Per effettuare l'operazione di scrittura si introduce il dato ed il suo complemento sulle linee di colonna (o di bit). Per la tecnologia I^2L (Integrated Injection Logic) il principio di funzionamento della cella elementare è identico a quello descritto in precedenza per le altre tecnologie.

Un fatto molto interessante è che la tecnologia I^2L, a differenza delle altre tecnologie menzionate in precedenza, offre la possibilità di realizzare delle memorie dinamiche, richiedenti cioè un rinfresco periodico di tutti i dati, come vedremo qui appresso.

In conclusione si può dire che le tecnologie bipolari per memorie statiche presentano le seguenti caratteristiche:

- elevata velocità di funzionamento
- compatibilità diretta DTL/TTL
- bassa capacità specie se confrontata con le memorie a MOS.

Memorie RAM a MOS.

Il rapido espandersi dei microprocessori, vale a dire di unità CPU realizzate su un solo chip, ha favorito nel campo dei MOS lo sviluppo di numerose soluzioni circuitali e tecnologiche che impiegano sia il principio di memorizzazione statica a bistabile sia il principio di memorizzazione dinamica.

Attualmente se si paragona la tecnologia MOS con la tecnologia bipolare si possono fare i seguenti rilievi di fondo:

- le memorie statiche a MOS raggiungono velocità comparabili a quelle realizzate con tecnologia bipolare;
- le memorie dinamiche a MOS raggiungono altissime densità ma presentano velocità inferiori a quelle ottenute con tecnologie bipolari.

Sebbene le memorie RAM a MOS possano essere fabbricate con tutti i tipi di tecnologia MOS, le tecnologie che consentono i migliori risultati sono le seguenti:

- Silicon-Gate Canale-N, nelle sue varie versioni;
- Ion-implantation di silicio policristallino;
- MOS complementari.

RAM statiche a MOS.

La cella elementare di memoria statica a MOS è classicamente un bistabile costituito da 6 transistori MOS, (fig. 3.11) e realizzata con tecnologia Silicon Gate Canale-N.

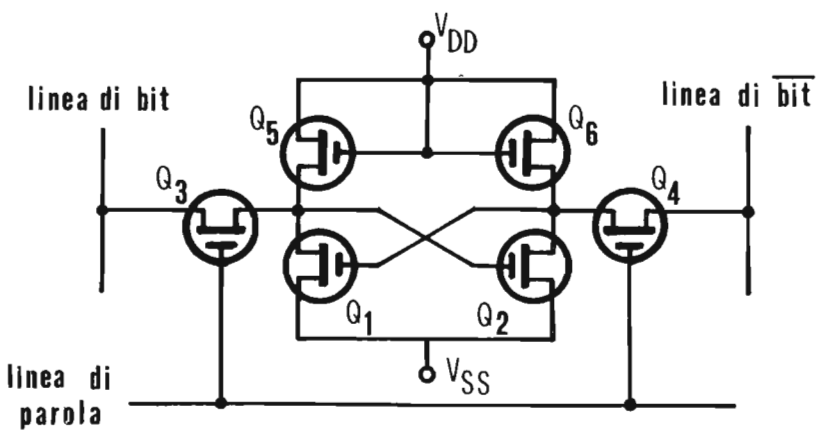


Fig. 3.11 — Cella elementare di memoria statica MOS realizzata a sei transistori e con tecnologia Silicen-Gate Canale-N.

I transistori Q_1 e Q_2 agiscono come elementi di memorizzazione, i transistori Q_5 e Q_6 esplicano la funzione di resistenza di carico, per cui talvolta possono essere di tipo depletion, altrimenti sono tutti di tipo enhancement, mentre i transistori Q_3 e Q_4 sono interruttori per l'accesso alle linee di bit.

Per le operazioni di scrittura/lettura, l'ingresso/uscita dei dati avviene sulla linea di bit e sulla sua complementare (linea di $\overline{bit}$), mediante il pilotaggio dei gates dei transistori Q_3 e Q_4 connessi alla riga di parola.

L'impiego della tecnologia ion-implantation di silicio policristallino, (fig. 3.12), permette la sostituzione dei transistori di carico Q_5 e Q_6 con resistenze vere e proprie, riuscendo in tal modo a migliorarne le velocità di funzionamento (paragonabile a quelle dei bipolari) e la densità di impaccamento.

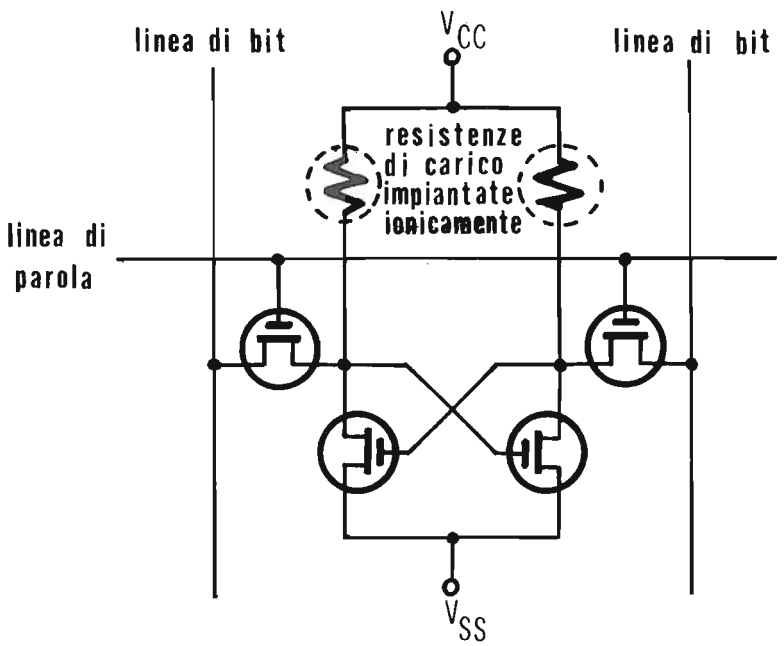


Fig. 3.12 — Cella elementare di memoria statica MOS con resistenze di carico realizzate mediante impiantamento ionico.

Una notevole riduzione della potenza dissipata, anche se la densità di impaccamento non è molto elevata, si ottiene utilizzando la tecnologia a MOS complementari.

Il funzionamento della cella a MOS complementari è del tutto analogo a quello già descritto per la cella a MOS classica, salvo che dal punto di vista costruttivo i transistori Q_1 e Q_2 sono a canale-N, mentre i transistori Q_3 e Q_4 sono a canale P (fig. 3.13).

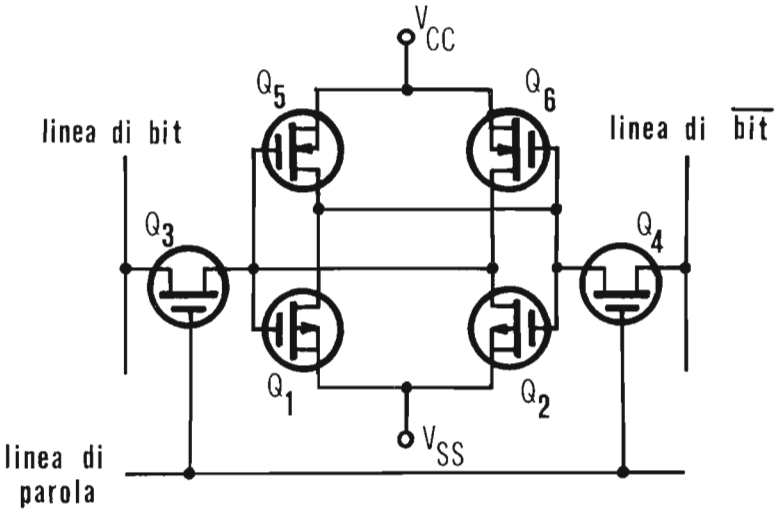


Fig. 3.13 – Cella elementare di memoria statica MOS realizzata con tecnologia CMOS (o MOS complementari).

RAM dinamiche a MOS.

Nella cella di memoria dinamica a MOS viene utilizzata quale elemento memorizzatore la capacità verso massa del gate: si comprende quindi la natura labile dell'elemento di memorizzazione per i fenomeni di scarica dovute alle correnti parassite e la necessità quindi di una rigenerazione o rinfresco periodico dei dati.

Nella corsa alla riduzione dell'area occupata da ogni cella elementare e nel tentativo di ridurre il numero dei contatti per migliorare le rese, sono state escogitate soluzioni che hanno progressivamente ridotto il numero dei

transistori MOS impiegati per la fabbricazione di una cella elementare dinamica a MOS.

Infatti i vari tipi di celle elementari dinamiche a MOS possono essere così suddivisi:

a) celle dinamiche a 4 transistori;
b) celle dinamiche a 3 transistori;
c) celle dinamiche a 1 transistore.

a) Cella dinamica a 4 transistori.

La cella dinamica a 4 transistori discende direttamente dalla semplificazione di una cella statica a 6 transistori, mediante la soppressione dei transistori MOS di carico Q_5 e Q_6 di fig. 3.11 e la memorizzazione dinamica dei dati sui transistori Q_1 e Q_2 (fig. 3.14), mantenuti nello stato voluto per effetto della carica immagazzinata nelle capacità di gate C_1 e C_2.

L'operazione di scrittura si effettua operando una selezione della riga di parola che pone in conduzione i transistori Q_3 e Q_4 ed introducendo il dato ed il suo complemento sulle rispettive linee di bit. Il dato presente sul nodo U passa al nodo U' e il complemento del dato presente sul nodo Z passa al nodo Z' producendo la carica e la scarica delle relative capacità C_1 e C_2.

Il procedimento di lettura è esattamente l'inverso di quello di scrittura in quanto il dato da leggere presente sul nodo U' ed il suo complemento presente sul nodo Z' vengono rispettivamente trasmessi ai nodi U e Z e quindi trasferiti alle linee di bit (o di colonna).

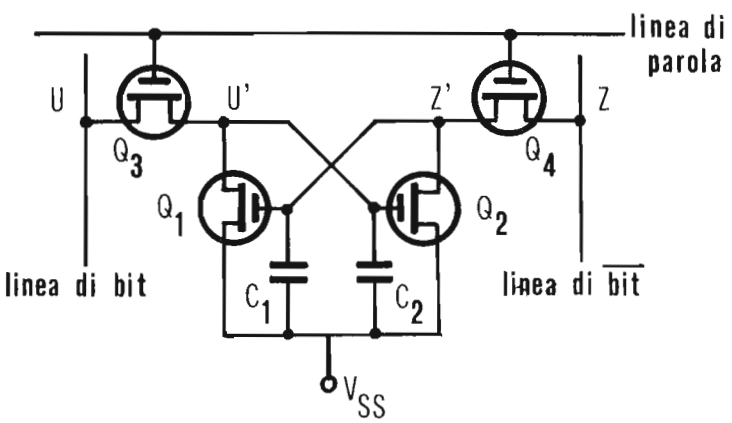


Fig. 3.14 — Cella elementare di memoria dinamica MOS a quattro transistori.

Per mantenere l'informazione si pongono periodicamente in conduzione i transistori Q_3 e Q_4 agendo sulla linea di parola mentre le linee di bit vengono portate ad uno stesso valore di tensione per permettere una rigenerazione della carica presente nelle capacità di memorizzazione, per effetto dell'accoppiamento incrociato gate-drenaggio dei due transistori MOS Q_1 e Q_2.

Questo tipo di cella è quello di più antica data e si presta ad essere realizzato con processi di tipo canale-P, ma non con canale-N per via delle elevate dimensioni che la cella elementare verrebbe ad assumere.

b) Cella dinamica a 3 transistori.

Una cella tipica di memoria a 3 transistori MOS è illustrata in fig. 3.15. La capacità C_S costituisce l'elemento fisico destinato all'immagazzinamento del dato.

L'operazione di scrittura inizia con l'applicazione di un segnale di abilitazione di scrittura sul gate del transistore Q_1 che si chiude mentre il transistore Q_2 è interdetto. Quindi il dato sulla linea di bit può essere trasmesso dal nodo U alla capacità C_S di memorizzazione del MOS Q_3. Alla fine della scrittura Q_1 viene nuovamente aperto, isolando la capacità C_S.

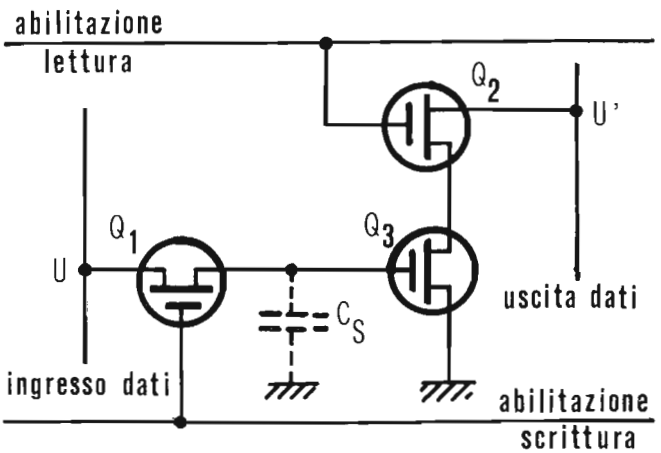


Fig. 3.15 — Cella elementare di memoria dinamica MOS a tre transistori.

L'operazione di lettura avviene portando in conduzione il transistore Q_2 mediante la selezione di una linea di parola di lettura, mentre il transistore Q_1 è interdetto.

Il dato memorizzato nella capacità C_S fluisce attraverso i transistori Q_3 e Q_2 fino al nodo U' per essere rivelato in uscita della linea di bit da un amplificatore di senso.

In questo tipo di memoria il rinfresco dei dati memorizzati è molto semplicemente realizzato leggendo ad intervalli regolari la cella e riscrivendo in essa l'informazione rivelata. Poiché esiste un amplificatore di senso distinto per ogni linea di bit (o di colonna) è possibile operare il rinfresco simultaneamente su tutte le celle appartenenti ad una stessa riga. Infatti se si fa riferimento alla fig. 3.16, basta attuare con un segnale di abilitazione lettura una delle M righe della matrice di memoria perché il contenuto di tutte le N celle della riga venga letta e trascritta nei rispettivi amplificatori di senso. Un successivo comando di abilitazione scrittura sulla stessa riga, permette di riscrivere i dati rigenerati dagli amplificatori di senso nelle rispettive celle della riga.

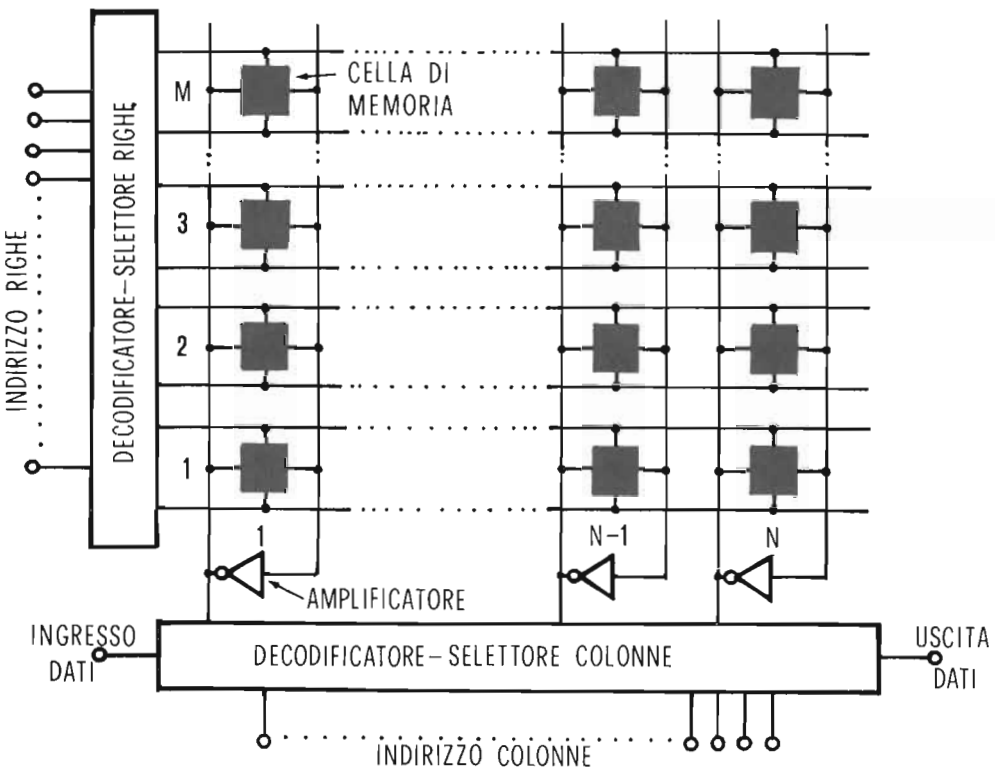


Fig. 3.16 — Schema per il rinfresco di una memoria dinamica con celle elementari a tre transistori.

c) Cella dinamica ad 1 transistore.

Si tratta della cella di memorizzazione più semplice possibile. Infatti consiste di un solo transistore MOS con in serie una capacità di memorizzazione. Il gate del transistore è collegato alla linea di parola mentre il drenaggio è collegato alla linea di bit. La linea di parola opera una selezione di riga mentre la linea di bit realizza la selezione di colonna congiuntamente alla funzione di linea dato. In effetti il transistore funziona da interruttore lasciando fluire le cariche dentro e fuori della capacità a seconda che si tratti di una operazione di lettura o di scrittura, (fig. 3.17).

Dal punto di vista costruttivo, l'area richiesta da questo tipo di cella è veramente esigua.

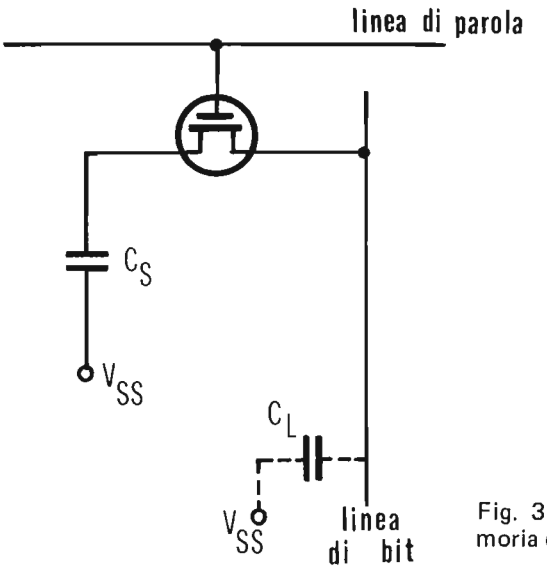


Fig. 3.17 – Cella elementare di una memoria dinamica MOS a un transistore.

Poiché la capacità C_S di memorizzazione è molto più piccola della capacità C_L della linea di bit, sorge il problema di avere, durante la lettura del dato, un segnale di ampiezza sufficiente alla discriminazione dei livelli "0" e "1".

Gli sforzi dei costruttori si sono dunque concentrati sul problema di aumentare la capacità di memorizzazione C_S e di diminuire quella di linea di bit C_L. In un procedimento costruttivo particolarmente interessante la capacità di memorizzazione ed il transistore MOS, sono affiancati direttamente l'uno all'altro; un elettrodo del condensatore di memorizzazione è

rappresentato dal substrato di silicio mentre l'altro elettrodo è ottenuto accrescendo una striscia di silicio policristallino ed il dielettrico è rappresentato dall'ossido di silicio intermedio. Con questo procedimento è stato possibile realizzare memorie a MOS dinamiche da 4096 bit (fig. 3.18).

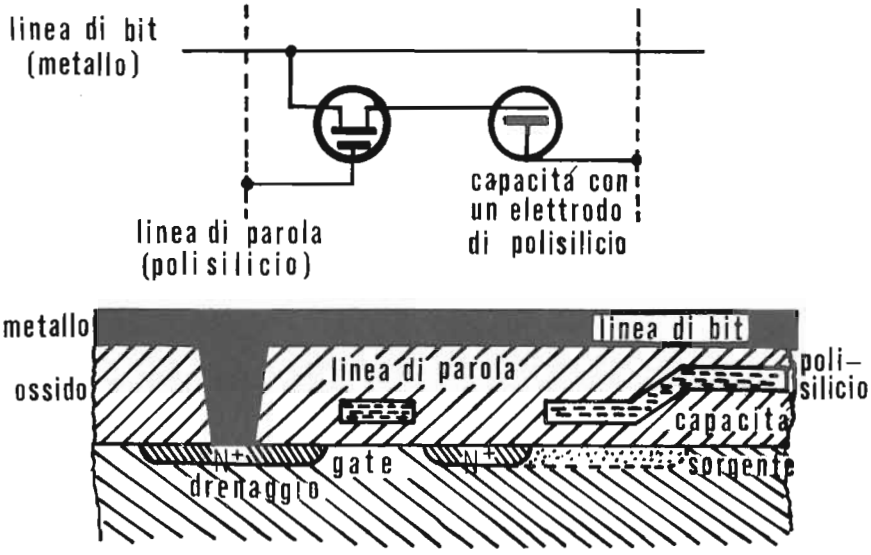


Fig. 3.18 — Realizzazione di una cella dinamica ad un transistore con procedimento a strato singolo di polisilicio.

Un'ulteriore riduzione di area si può ottenere con il procedimento a "doppio strato di polisilicio". In questa soluzione costruttiva il transistore MOS è sovrapposto alla capacità di memorizzazione, mediante l'utilizzazione di due livelli di interconnessione. Poiché non è più necessaria alcuna area diffusa (sorgente del MOS) fra il gate del MOS e la capacità di memoria, (fig. 3.19), si ha una notevole riduzione dell'area occupata da ogni cella e si possono realizzare in tal modo memorie dinamiche da 16.384 bit.

In questi tipi di cella ogni operazione di lettura è distruttiva per cui occorre riscrivere ogni volta il dato letto. Se si strutturano opportunamente gli amplificatori di senso posti in uscita delle linee di bit (un amplificatore per ogni linea di bit) l'operazione di scrittura dei dati appena letti non presenta difficoltà così come l'operazione per il rinfresco dei dati memorizzati, che può avvenire contemporaneamente su tutte le celle appartenenti a una stessa riga.

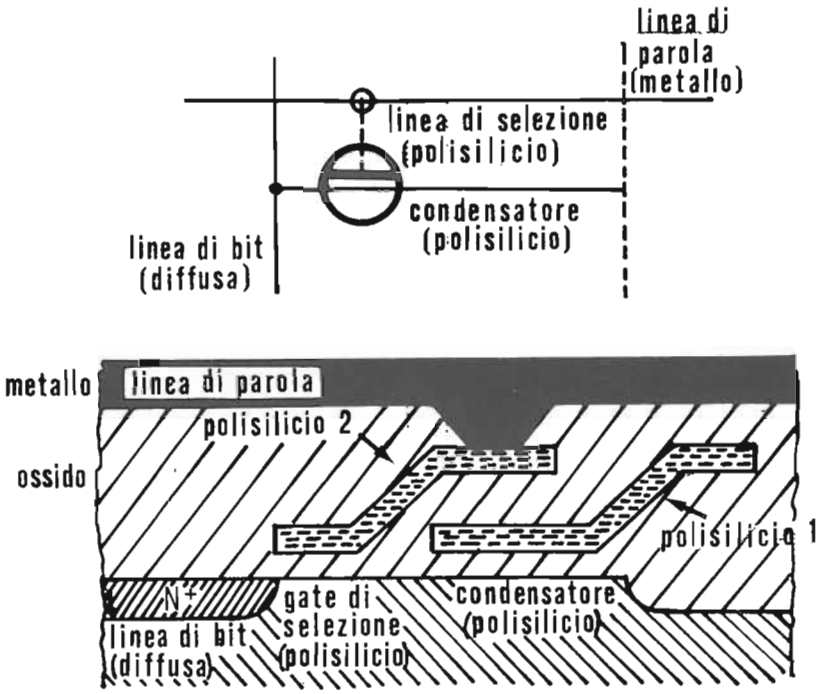


Fig. 3.19 — Realizzazione di una cella dinamica ad un transistore con procedimento a strato doppio di polisilicio.

3.9.3. Le memorie ROM a semiconduttore.

Le memorie ROM (Read Only Memory) o memorie di sola lettura costituiscono un caso particolare delle memorie RAM in quanto ogni cella di memoria ROM contiene un dato fisso, determinato all'atto della fabbricazione della memoria e non modificabile mediante una operazione di scrittura. Si possono quindi eseguire su tali memorie soltanto operazioni di lettura.

Memorie ROM bipolari.

La cella elementare di memoria ROM bipolare impiega una struttura basata su un transistore a multiemettitore, pilotato sul collettore dal decodificato di riga, mentre i vari emettitori rappresentano le colonne della ROM (fig. 3.20).

Un emitter di tale transistore rappresenta il dato memorizzato, di modo che durante la mascheratura del dispositivo basta costruire o non costruire l'emettitore corrispondente per avere la presenza o l'assenza del dato nella cella elementare.

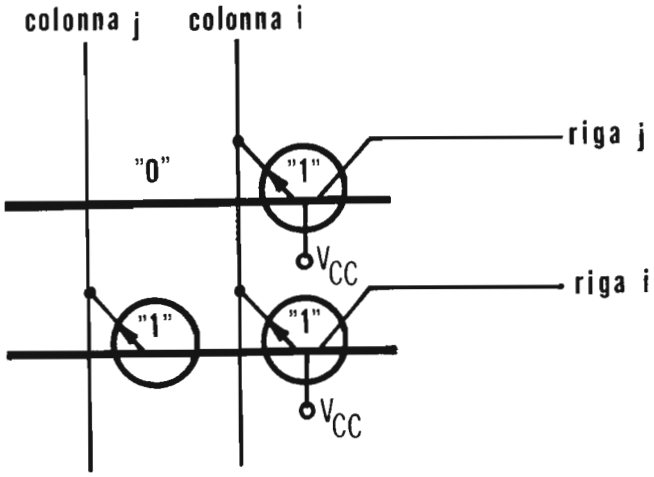


Fig. 3.20 — Cella di memoria ROM bipolare con transistore multiemettitore: la presenza o l'assenza di un emettitore rappresenta il contenuto fisso della cella.

Di norma si ricorre alle ROM bipolari quando occorre un'elevata velocità di funzionamento anche a scapito di una ridotta capacità di memorizzazione.

Memorie ROM a MOS.

La cella elementare di memorie ROM a MOS impiega un semplice transistore MOS, pilotato sul gate dal decodificatore di riga mentre il drenaggio è collegato alla colonna della ROM (fig. 3.21).

La presenza o l'assenza del transistore MOS nella cella elementare determina la memorizzazione o meno di un dato.

Le tecnologie più frequentemente impiegate per la fabbricazione di ROM a MOS sono quelle a Silicon Gate e quelle a Metal Gate.

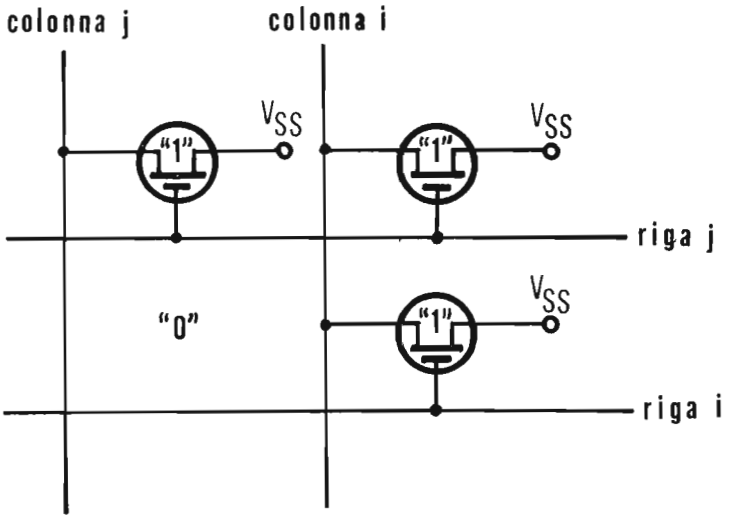


Fig. 3.21 — Cella di memoria ROM a MOS: la presenza o l'assenza di un transistore MOS rappresenta il contenuto fisso della cella.

Memorie PROM (Programmable ROM).

Nelle memorie ROM la definizione delle informazioni contenute avviene durante la fabbricazione del dispositivo mediante un'operazione di mascheratura. Questo procedimento può comportare costi elevati per l'utilizzatore se si tratta di un numero limitato di esemplari.

Le memorie PROM ovviano a questo inconveniente in quanto il contenuto della memoria può essere stabilito presso l'utilizzatore anziché presso il fabbricante durante la fabbricazione.

La cella elementare di una PROM a tecnologia bipolare è costituita da un transistore multiemettitore pilotato sul collettore e con tutti gli emettitori collegati in partenza alle colonne corrispondenti mediante una metallizzazione fusibile realizzata con leghe speciali (fig. 3.22). Perciò inizialmente tutte le celle della memoria contengono "1". Laddove occorre scrivere "0", si esegue sulla memoria un ciclo particolare su una apparecchiatura operante a tensioni superiori a quelle di funzionamento che permettono la fusione del collegamento fusibile con conseguente esclusione dell'emettitore della cella interessata, che ad un'interrogazione risponderà con "0".

Il tipo di scrittura descritto è irreversibile e quindi una PROM è utilizzabile per un solo pattern.

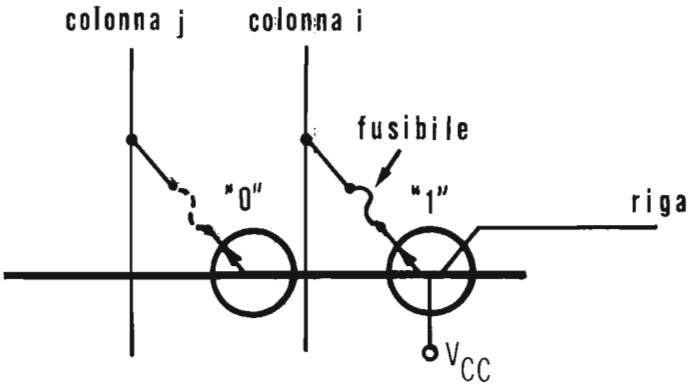


Fig. 3.22 – Cella elementare di memoria PROM bipolare: il fusibile intatto o fuso rappresenta il contenuto fisso della memoria.

Memorie EPROM (Erasable PROM).

Poiché le PROM sono utilizzabili per un solo pattern e si possono costruire solo con tecnologia bipolare, i costruttori sono ricorsi alle tecnologie MOS per realizzare delle PROM cancellabili e quindi riusabili.

La cella elementare di una memoria EPROM è costituita da un MOS particolare, in grado di immagazzinare, entro l'ossido di gate, per un tempo estremamente lungo, le cariche relative allo stato logico che si vuole memorizzare. Il gate del MOS non è collegato a nessun'altra parte del circuito (gate flottante) (fig. 3.23).

In partenza tutte le celle sono nello stato "0", cioè il gate flottante è scarico. Quando si realizza un particolare ciclo a tensioni superiori a quelle tipiche di funzionamento, si produce una trasmigrazione di cariche attraverso l'ossido che si fissano nel gate flottante corrispondente alle celle dove si vuole memorizzare "1".

Per poter utilizzare la memoria per altre applicazioni, si cancella il contenuto esponendo il chip a luce ultravioletta che provvede a distruggere le cariche memorizzate.

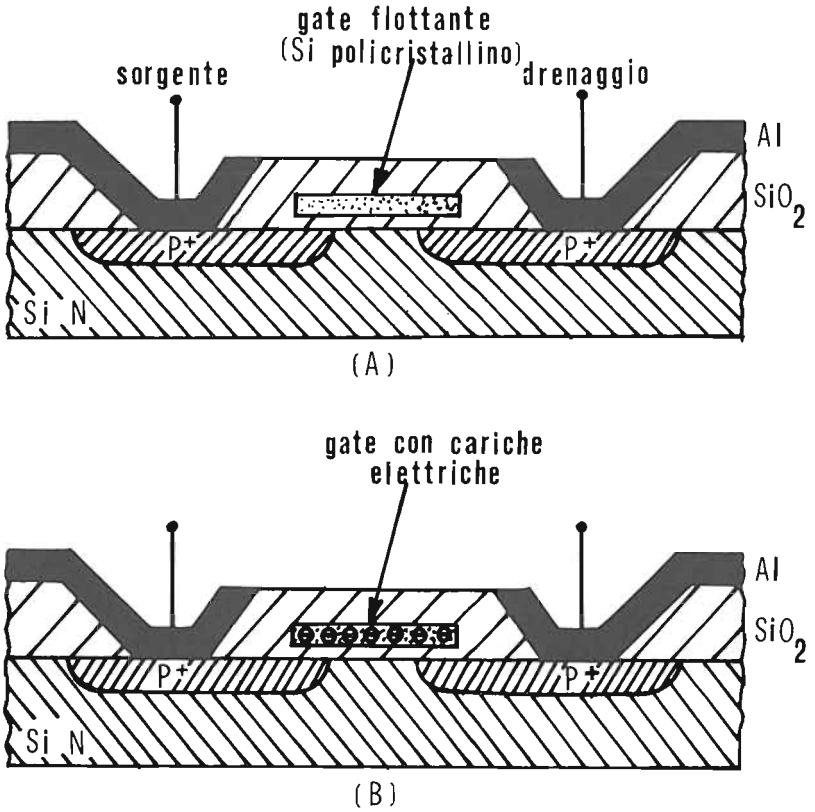


Fig. 3.23 – Cella elementare di memoria EPROM:
(A) - Cella contenente "0" (non programmata o cancellata) perché il gate flottante è privo di cariche elettriche.
(B) - Cella contenente "1" (programmata) perché il gate flottante è carico.

3.10. LE MEMORIE A TRASFERIMENTO DI CARICA (CCD-Charge Coupled Devices).

3.10.1. Generalità.

Le memorie a trasferimento di carica (CCD) sono costituite da una struttura MIS (Metallo - Isolante - Semiconduttore) estremamente semplice: un

convenzionale substrato di silicio ricoperto da un sottile strato isolante, sul quale sono depositati degli elettrodi metallici, disposti in modo regolare.
La struttura MIS è rappresentata in fig. 3.24.

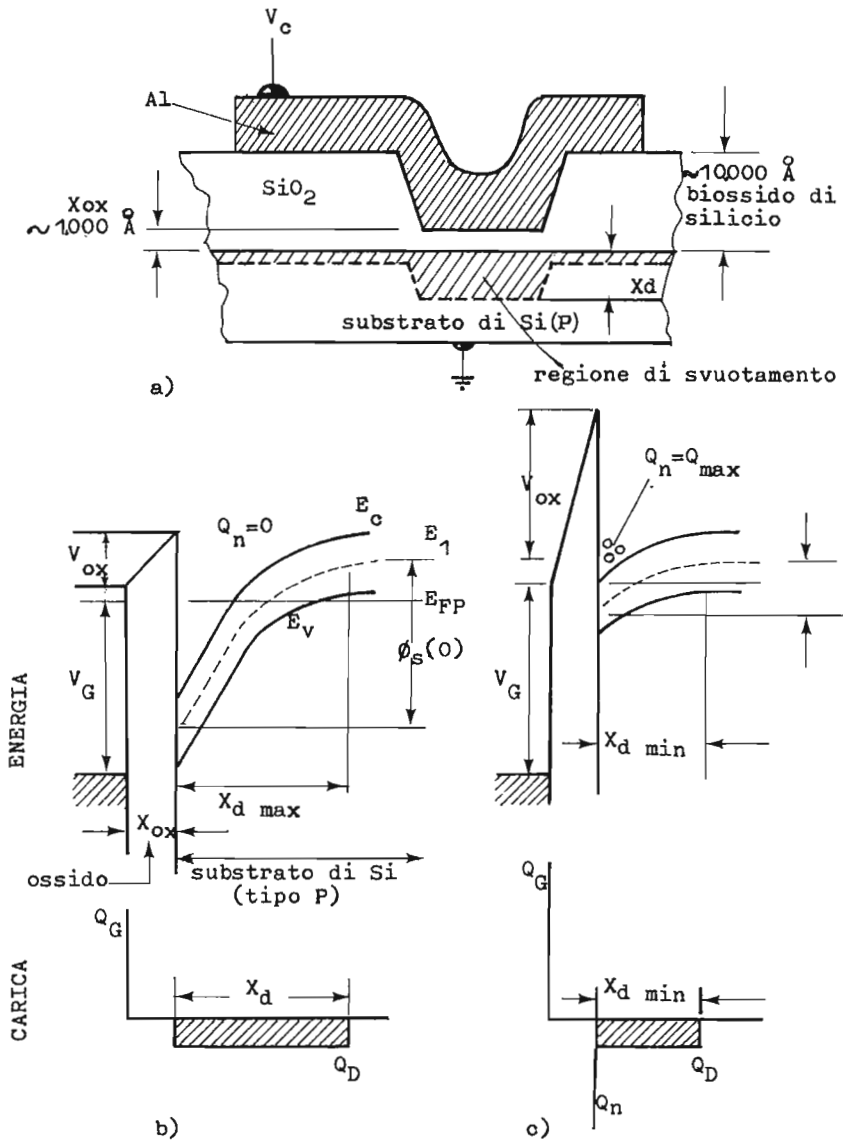


Fig. 3.24 — Struttura MIS usata nei dispositivi CCD.
L'informazione è immagazzinata in forma di carica elettrica nella regione di svuotamento.

Applicando una tensione opportuna al gate metallico (positiva nel caso della fig. 3.24 in cui il substrato di Si è drogato P), si ottiene una zona di svuotamento sotto il gate stesso, nel Si.
Non appena la tensione viene applicata si crea un profilo delle bande di energia come in fig. 3.24 (b) dove si nota una forte caduta di tensione nel silicio (ϕ_S, che chiamiamo tensione superficiale) e una forte zona di potenziale inversione. Gli elettroni liberi delle zone circostanti tendono a fluirvi; però, in assenza di zone adiacenti drogate N, il flusso è dato soltanto dagli elettroni di origine termica (Dark current) ed è molto piccolo.

Dopo un certo tempo la zona si riempie comunque di elettroni e si perviene alla situazione di equilibrio termico rappresentata nella fig. 3.24 (c). Si ha una zona superficiale di forte inversione, che diminuisce molto la tensione superficiale ϕ_S, aumentando nel contempo la caduta nell'ossido. Questa situazione si raggiunge in un tempo T_R, tempo di rilassamento termico; i valori di T_R possono variare da 1 sec. a qualche minuto.

In fig. 3,25 è riportata ancora la struttura MIS con i profili della tensione superficiale ϕ_S.

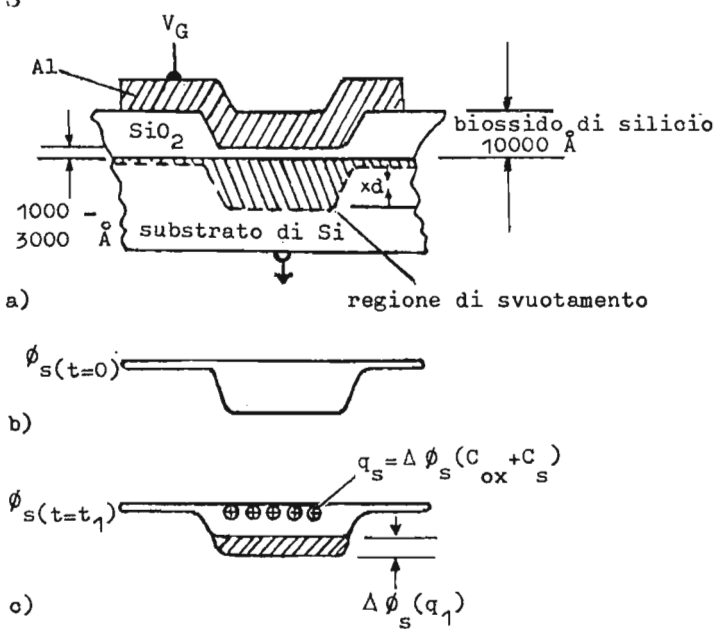


Fig. 3.25 — a) Sezione di una struttura MIS; b) profilo della tensione superficiale appena dopo l'applicazione di un gradino di tensione V_G; c) profilo della tensione superficiale con una carica-segnale q_S nel pozzo di potenziale.

Sotto l'elettrodo con tensione applicata, si forma un "pozzo di potenziale", il quale, ricorrendo a una analogia idraulica, può essere o no riempito di cariche elettriche, allo stesso modo in cui un contenitore può essere riempito da un liquido.

Per tempi molto inferiori al tempo di rilassamento (es. 1/1000) il pozzo di potenziale resta praticamente vuoto di cariche di origine termica, e quindi atto a ricevere e immagazzinare una carica trasmessagli in qualche modo.

In fig. 3.25(b) è indicato il profilo di ϕ_S con il pozzo vuoto di cariche: in fig. 3.25(c) invece è riportato il profilo di ϕ_S con una certa carica intrappolata.
La carica agisce nel senso di diminuire la tensione superficiale e quindi nell'analogia idraulica è come se alzasse il livello.
Una struttura MIS è quindi adatta a immagazzinare un bit sotto forma di carica elettrica (presenza carica = "1", assenza carica = "0") per un tempo limitato, dell'ordine di qualche msec.

La cella base CCD consiste di alcune strutture MIS adiacenti, lungo le quali la carica elettrica viene spostata, inviando tensioni opportunamente sfasate sui relativi elettrodi.

3.10.2 Principio di funzionamento di una memoria a trasferimento di carica.

La cella elementare di memoria CCD che è stata descritta nel paragrafo precedente può essere realizzata mediante una terna di elettrodi adiacenti. Ad ogni istante, una sola delle posizioni è effettivamente usata per immagazzinare l'informazione; le altre due servono per trasferirla, evitando interferenze con le celle adiacenti (fig. 3.26).

Le sequenze necessarie per produrre uno spostamento dell'informazione possono essere così schematizzate.

In primo luogo si applica una tensione negativa al substrato di silicio, di tipo N, in modo da formare uno strato di svuotamento uniforme sotto tutti gli elettrodi. Al primo elettrodo ϕ_1 della cella viene poi applicata una tensione negativa superiore a quella applicata per formare uno strato di svuotamento uniforme sotto tutti gli elettrodi, in modo da creare sotto il primo elettrodo una regione di svuotamento più profondo, che spazialmente definisce una buca di potenziale. Questa buca di potenziale è in grado di attrarre e di trattenere le cariche (costituite da portatori minori-

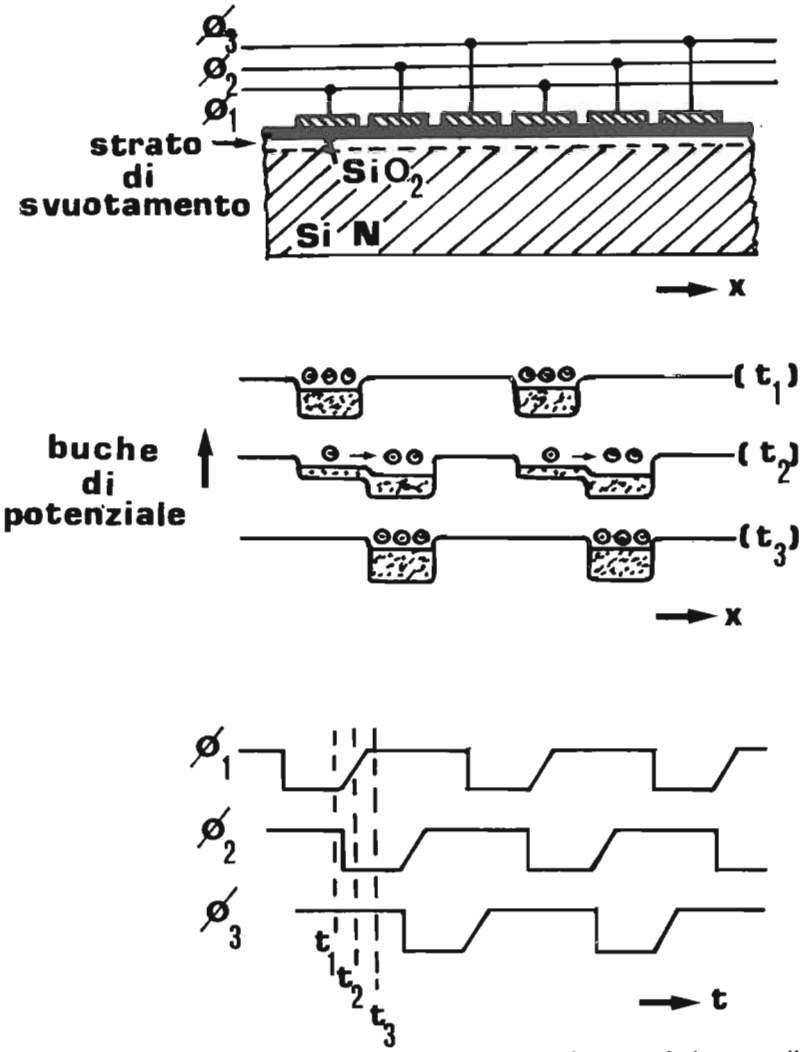


Fig. 3.26 – Principio di funzionamento di una memoria a trasferimento di carica (CCD).

tari o lacune) che possono essere iniettate nel substrato in vari modi.
La presenza o l'assenza di questa carica corrisponde a "1" o a "0".

Per trasferire la carica immagazzinata, si applica poi una tensione negativa al secondo elettrodo ϕ_2 della cella in modo da creare una buca di potenziale sotto questo elettrodo, più profondo di quella generata sotto il primo elettrodo.
Le cariche trattenute sotto il primo elettrodo fluiranno quindi sotto il

secondo. Analogo trasferimento viene attuato applicando un impulso negativo al terzo elettrodo ϕ_3. Terminata così la sequenza completa di impulsi, la carica si trova spostata di uno stadio (tre elettrodi). Essendo la struttura geometricamente simmetrica, l'informazione memorizzata può essere trasferita nei due sensi.

Se si rinuncia alla possibilità di trasferimento bidirezionale, si possono allora realizzare delle strutture asimmetriche che consentono di ridurre a due il numero di fasi.

La generazione delle cariche che rappresentano l'informazione viene effettuata nel modo seguente. Facendo riferimento alla fig. 3.27, la diffusione P a sinistra e l'elettrodo G formano una sorgente MOS di corrente; un impulso di tensione applicato all'ingresso dati G provoca un flusso di corrente nella buca di potenziale sotto il primo elettrodo ϕ_1. La rivelazione delle cariche che rappresentano l'informazione viene analogamente effettuata trasferendo la carica in una diffusione P che scarica la struttura MOS, come indicato in fig. 3.27 a destra.

Poiché le cariche minoritarie iniettate nella regione di svuotamento non hanno permanenza indefinita in quanto tendono a annichilarsi con i portatori maggioratori presenti nel substrato di silicio, occorre rinfrescare il contenuto della memoria entro tempi dell'ordine di qualche centinaio di microsecondi. Questo tipo di memoria ha quindi un funzionamento dinamico con circolazione continua dell'informazione. Ne consegue che l'accesso nella memoria CCD è seriale. In pratica però la memoria viene organizzata in blocchi di registri a scorrimento di lunghezza fissa, ognuno dei quali accessibile a caso.

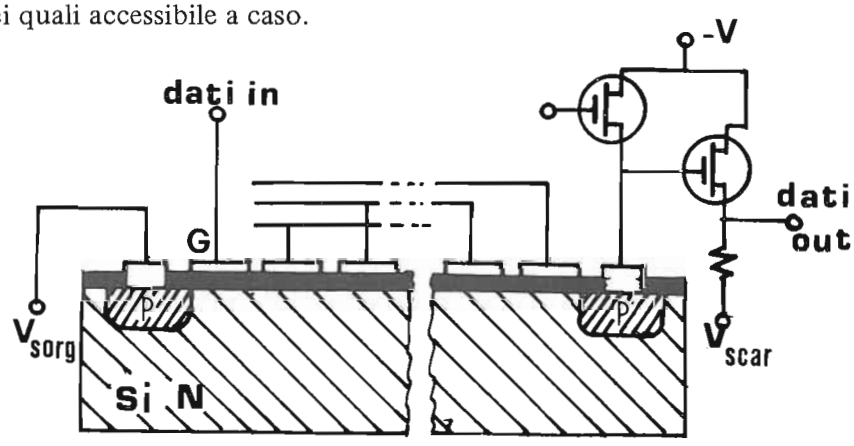


Fig. 3.27 - Stazione di generazione e di rivelazione dei dati in un dispositivo CCD.

Rispetto alle memorie RAM costruite con tecnologia a semiconduttore LSI, la memoria CCD presenta una particolare semplicità strutturale, derivante dal fatto che la carica rimane sempre nel substrato del dispositivo per cui non è richiesta né la realizzazione di geometrie complicate come nel caso delle strutture integrate convenzionali né l'impiego di processi critici come la diffusione di impurità nel semiconduttore.
Si può ritenere che la densità dell'informazione per unità di superficie della memoria CCD resterà da 3 a 5 volte superiore a quella delle memorie RAM dinamiche a semiconduttore. Però, essendo l'accesso solo parzialmente a caso (cioè accesso a caso ad un registro e accesso seriale all'informazione contenuta nel registro stesso) la velocità della memoria CCD risulta minore di quella delle RAM a semiconduttore.

Si pensa che le memorie CCD siano suscettibili ancora di grandi progressi. Attualmente sono disponibili chip di non elevata capacità, 16K bit al massimo, con tempo medio di accesso di 100 μsec e velocità di trasferimento dell'informazione di 2 M bit/sec.

3.11. LE MEMORIE A BOLLE MAGNETICHE.

3.11.1. Generalità.

Le memorie che vengono descritte in questo paragrafo differiscono per numerosi aspetti dalle memorie realizzate fino ad ora con materiali magnetici. Qualsiasi memoria che impieghi un materiale magnetico come supporto di memorizzazione sfrutta il fatto che un materiale magnetico può assumere due stati di equilibrio molto stabili, facilmente discriminabili tra di loro e capaci di permanere indefinitivamente nel tempo senza alcun consumo di energia.

Le memorie a bolle magnetiche fanno ricorso ad uno strato di opportuno materiale ferro o ferri-magnetico che presenti una direzione di facile magnetizzazione (o anisotropia) in direzione normale allo strato, di modo che la magnetizzazione risulti sempre perpendicolare al piano del substrato, con direzione verso l'alto o verso il basso. In assenza di campo magnetico esterno il film presenta un assetto a domini magnetici, dove domini adiacenti hanno verso di magnetizzazione opposto e perpendicolare al piano dello strato. L'assetto dei domini corrisponde ad una situazione di minima energia. (fig. 3.28 (a)).

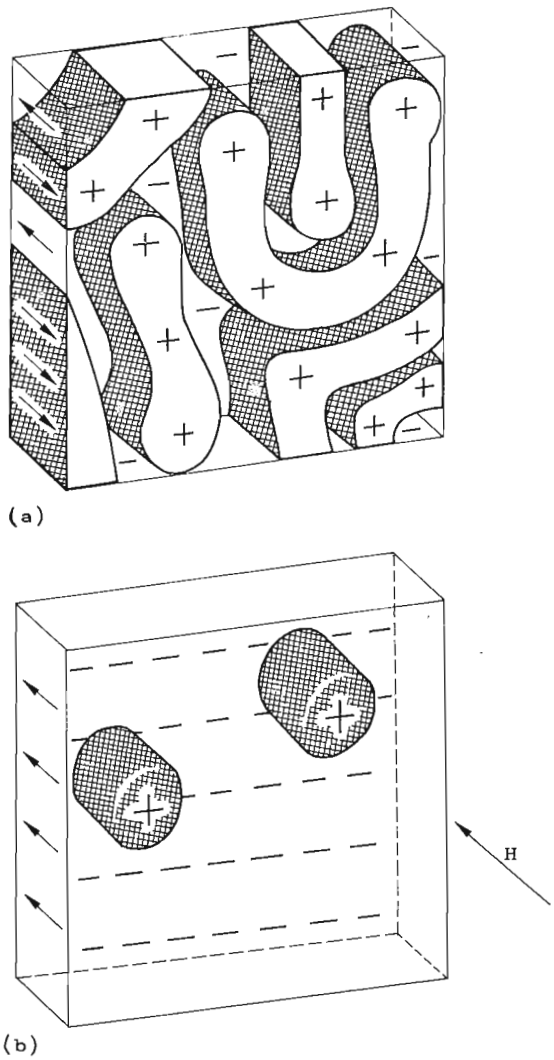


Fig. 3.28 – Domini magnetici nelle ortoferriti.

(a) Senza campo di polarizzazione. (b) Con campo di polarizzazione.

Se si applica un campo magnetico esterno perpendicolare al piano del film, i domini orientati nel verso del campo accrescono di dimensioni, mentre quelli orientati nel senso opposto diminuiscono; al crescere del campo questi ultimi si trasformano in isole cilindriche, il cui diametro di base è proporzionale all'intensità del campo. Questi domini cilindrici prendono il nome di "bolle magnetiche" (fig. 3.28 (b)). Aumentando ancora l'inten-

sità del campo esterno, le bolle si restringono fino a svanire per un dato valore del campo esterno: tutto lo strato è allora polarizzato nella direzione del campo. Quindi per un dato tipo di materiale, le bolle magnetiche sussistono stabilmente, una volta create, in un preciso intervallo di valore del campo magnetico esterno.

Ora, la presenza o l'assenza di bolle in determinate posizioni del substrato può essere utilizzata per rappresentare informazioni binarie.
Non solo, ma le bolle possono muoversi nel piano dello strato in qualunque direzione, sotto l'azione di un campo parallelo al piano del film.

3.11.2. Principio di funzionamento di una memoria a bolle magnetiche.

Per far rimanere le bolle sul substrato, si traccia il percorso da seguire depositando sulla superficie del substrato stesso una sequenza di elementi geometrici realizzati con materiale ad alta permeabilità come il permalloy.

La geometria più diffusa è costituita da una trama regolare di barrette a I e a T.

La presenza di un campo magnetico parallelo al piano del film induce una polarizzazione delle barrette depositate di permalloy che attirano le bolle verso il proprio polo negativo, se si suppone che le bolle abbiano il proprio polo positivo rivolto verso l'alto. (fig. 3.29).

Al termine di un giro completo del campo magnetico rotante le bolle si sono spostate da una barra a T a quella successiva.

Si pongono a questo punto due problemi: come generare una bolla in una certa posizione iniziale e come rivelarla in una data posizione terminale.

Per la generazione vengono usati vari metodi, tra cui i principali sono i seguenti. Un metodo detto di duplicazione da una bolla germe, consiste nell'estrarre per duplicazione una bolla da una sorgente o germe che contiene sempre una bolla, come indicato in fig. 3.30. La sorgente è una zona ricoperta da un dischetto di permalloy. Il campo magnetico rotante provoca un allungamento della bolla germe fino al punto di spezzarla in due. Tale operazione può essere favorita da un colpo di corrente in un anello conduttore che annulla il dominio nella zona di taglio. La bolla è generata quando si vuole inserire un "1" nella memoria.
Per inserire uno "0" occorre bloccare la generazione della bolla; ciò si ottiene mediante un conduttore di inibizione depositato sul substrato, in-

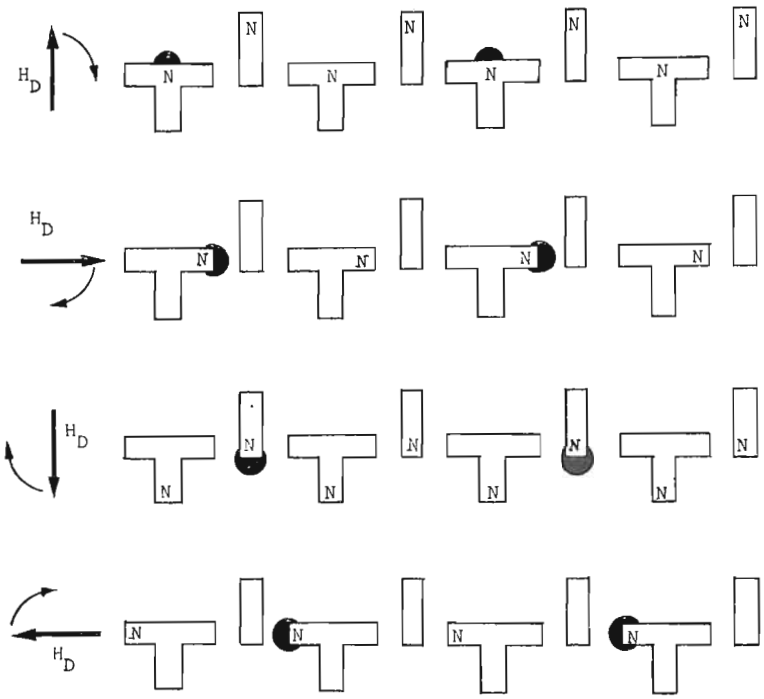


Fig. 3.29 – Spostamento delle bolle con barre a T e campo magnetico rotante nel piano del film.

viando corrente sul quale si crea un campo che contrasta l'azione del campo rotante sulla bolla germe. Un generatore di questo tipo opera con frequenze limitate.

Un altro metodo che dà migliori prestazioni consiste nel nucleare la bolla quando occorra scrivere un "1". La nucleazione avviene in una zona in cui sono alterate le proprietà del granato, e precisamente vi è stata abbassata l'anesotropia del film mediante un riscaldamento localizzato con raggio laser.

La bolla si forma solo in questa zona inviando un impulso di corrente in una spira conduttrice che sovrasta l'areola trattata termicamente.

Per la rivelazione delle bolle vi sono vari sistemi riconducibili a tre categorie:

1) lettura a induzione: la tensione indotta dal passaggio di una bolla in una spira depositata sul substrato viene amplificata fino ad ottenere un segnale logico.

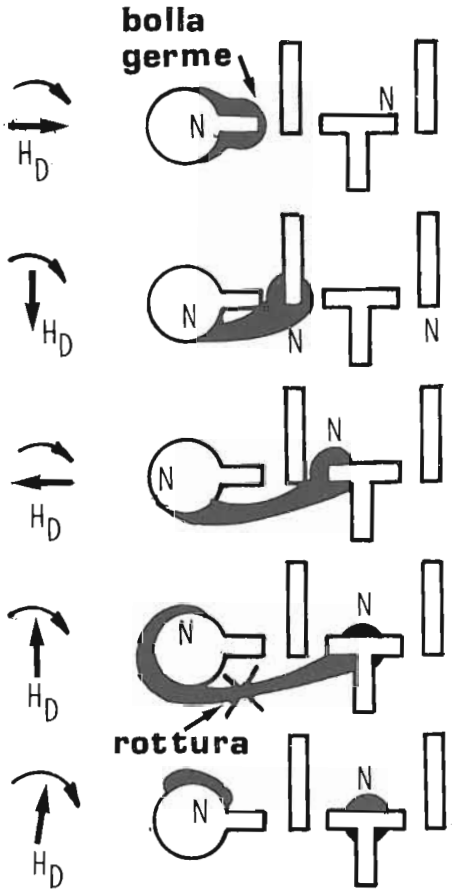
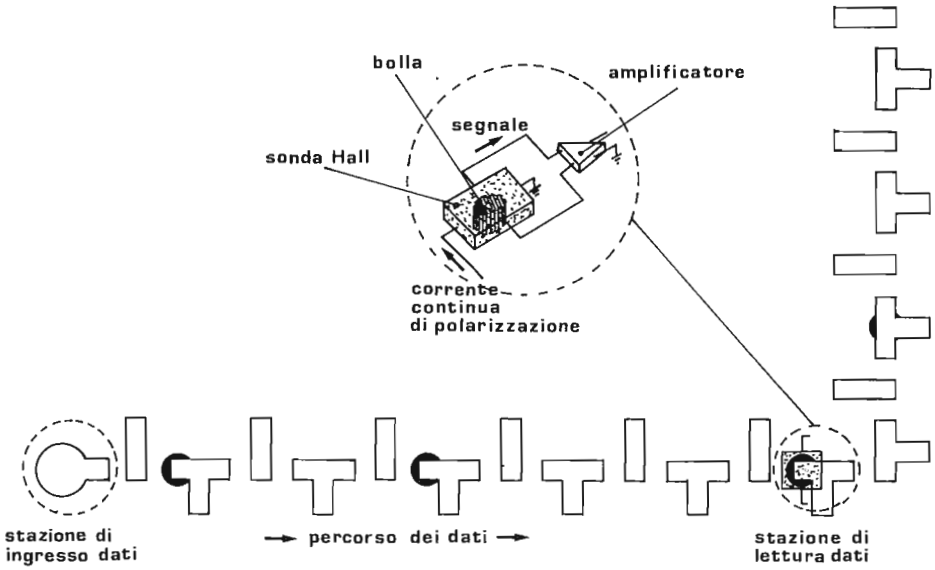


Fig. 3.30 — Metodo per la generazione di bolle magnetiche.

2) lettura a magnetoresistenza: la variazione della resistenza provocata in una sonda magnetiresistiva dal campo magnetico della bolla da rivelare viene amplificata e manipolata fino ad ottenere un segnale logico.
3) lettura per effetto Hall: si tratta di uno dei sistemi preferiti anche se richiede la deposizione di un materiale appropriato che costituisce la sonda ad effetto Hall (fig. 3.31).

Per quel che riguarda i metodi di organizzazione logica di un chip di memoria a bolle, descriveremo il metodo più usato detto ad "anello principale-secondario". Lo schema di principio dell'organizzazione ad anello principale-secondario è riportato in fig. 3.32.

Fig. 3.31 — Rivelazione di bolle magnetiche mediante sonda ad effetto Hall.



Tale sistema consiste nell'immagazzinare le informazioni in un certo numero di anelli detti secondari. Gli anelli secondari fanno capo tutti a un anello principale il quale funge da collegamento tra gli anelli secondari e le stazioni di scrittura e di lettura.

Ogni anello secondario fa capo all'anello principale mediante una stazione di trasferimento costituita da un commutatore bidirezionale che viene attivato inviando una corrente nel conduttore di trasferimento. Tale conduttore comanda tutti i commutatori: in lettura un bit viene trasferito da ciascun anello secondario sull'anello principale; in scrittura il trasferimento avviene in senso opposto.

Il senso in cui avviene il trasferimento è determinato dall'orientamento del campo magnetico rotante al momento dell'invio della corrente.

I bit trasferiti sull'anello principale in lettura vengono poi spostati e letti serialmente sulla stazione di lettura; per mantenere l'informazione occorre ritrasferirla negli anelli secondari; la bolla quindi oltrepassa la stazione di lettura, viene fatta ritornare sul suo anello e ritrasferita. Nel caso si voglia scrivere si azzerano i bit trasferiti sull'anello maggiore e lo si carica con i bit provenienti dalla stazione di scrittura.

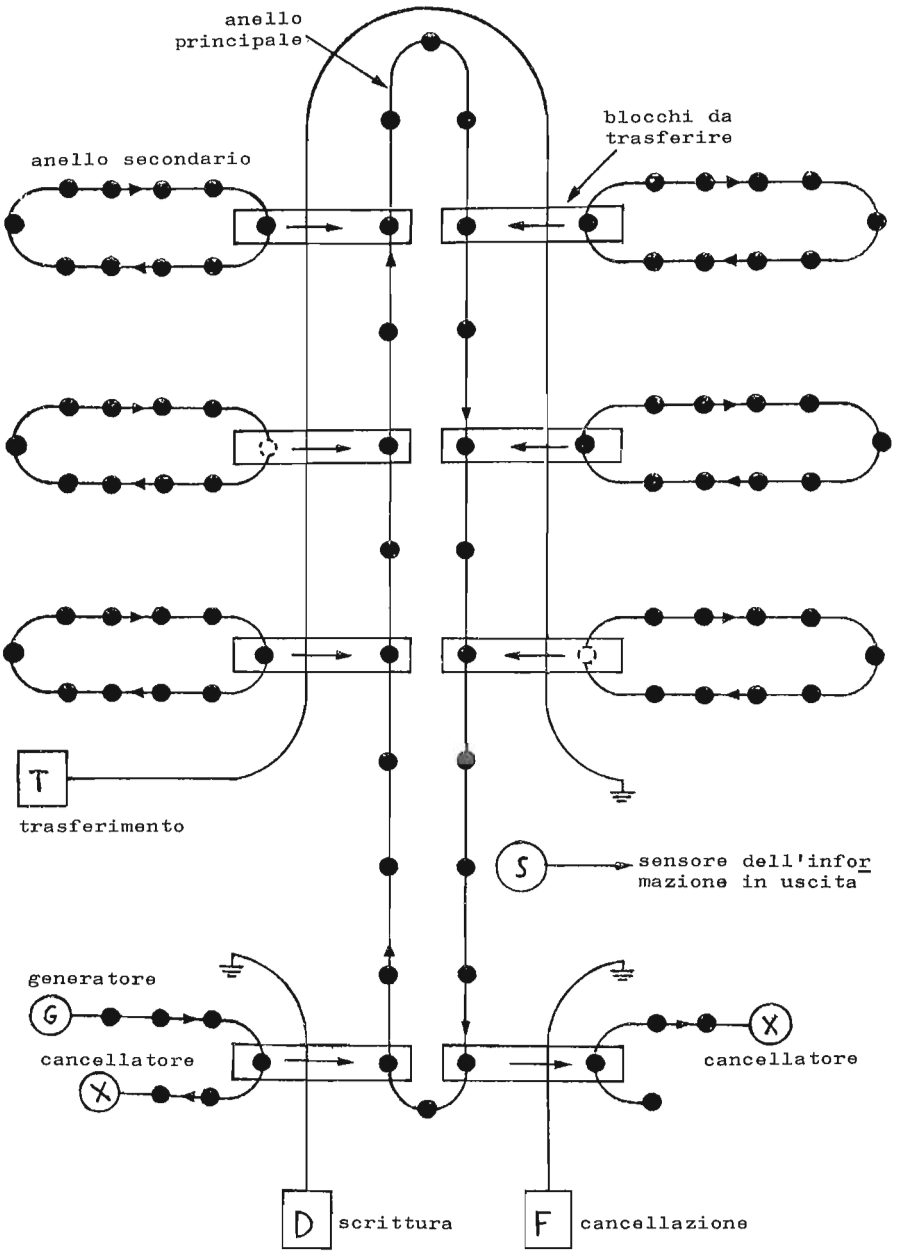


Fig. 3.32 — Organizzazione di un chip ad anello principale-secondario per memoria a bolle magnetiche.

Le operazioni di lettura e scrittura avvengono sempre per blocchi di bit pari al numero di anelli secondari.
I bit di un blocco sono quelli aventi eguale posizione sui vari anelli secondari. Tale organizzazione consente di ottimizzare il rapporto costo/prestazioni.

Infatti la detezione del segnale in uscita richiede un amplificatore esterno abbastanza costoso, il quale deve scrivere un numero sufficiente di bits (es. 16K).
D'altro lato interessa poter accedere rapidamente (senza spostare serialmente tutti i 16 Kbit) a un blocco limitato di informazioni (es. 64 - 128 o 256 bit). Con l'organizzazione anello principale-secondario l'accesso al blocco è poco maggiore del numero di bit presenti nell'anello secondario per il tempo di spostamento di una posizione.
Il funzionamento di questo schema è basato sul trasferimento di bolle da un anello all'altro.

L'informazione letta non è distrutta ma viene parcheggiata nell'anello principale: deve essere comunque rimessa negli anelli secondari dopo la lettura.
Sono stati attuati altri schemi dove le bolle uscenti dall ' anello secondario vengono duplicate: una circola nell'anello secondario. l'altra viene immessa nell'anello principale. In tal caso occorre un dispositivo duplicatore/annullatore alla congiunzione anello principale-secondario: l'informazione circola nell'anello secondario,non ha bisogno di essere riscritta, ma deve essere annullata prima della scrittura.

Il problema principale delle memorie a bolle è costituito dal materiale del substrato. Questo deve possedere, come si è visto, un'asse facile di magnetizzazione perpendicolare al suo piano. Oltre a ciò, si richiedono elevate doti di purezza e di perfezione del materiale, perché i difetti interferiscono con la mobilità delle bolle, riducendo o abolendo la loro capacità di trasferire l'informazione.

I materiali oggi impiegati sono sia monocristallini (appartenenti alle famiglie delle ortoferriti e dei granati), sia amorfi (costituiti da leghe metalliche diverse); questi ultimi possono essere depositati su substrati di vario tipo e sono più facili da ottenere dei primi.
La densità di informazioni nelle memorie a bolle è funzione principalmente delle dimensioni delle bolle. Attualmente queste hanno diametri dello

ordine di qualche micron; in laboratorio si sono però ottenuti valori inferiori al micron.

Oltre al diametro delle bolle, un altro fattore che limita la densità delle informazioni è costituito dalle interazioni fra le bolle medesime. Queste si comportano infatti come piccoli magneti e cominciano a disturbarsi allorché la loro distanza diventa dell'ordine di alcune volte il loro diametro; nella memoria esse debbono pertanto essere opportunamente distanziate (4-5 diametri).

Da un punto di vista circuitale e sistemistico, sia le memorie CCD che le memorie a bolle magnetiche danno luogo ad organizzazioni a registri di scorrimento.

Le bolle magnetiche hanno il pregio essenziale della permanenza delle informazioni; operano però a velocità relativamente basse; le memorie CCD consentono invece prestazioni elevate, ma esigono un periodico rinfresco dell'informazione, con frequenza piuttosto alta.

Dal punto di vista tecnologico le CCD non presentano difficoltà sostanziali, perché sono dispositivi a semiconduttore, che sfruttano le tecniche costruttive lungamente collaudate dei MOS.
Le bolle magnetiche invece rappresentano una tecnologia completamente nuova, con grossi problemi di base.
E' presumibile che potranno sussistere entrambe con differenti campi di applicazione. Per quanto riguarda la non volatilità (permanenza delle informazioni a macchina spenta) occorre notare che le memorie CCD sono volatili: per conservare l'informazione non solo occorre tenere le tensioni continue ma anche far ciclare la memoria a una frequenza minima.
Le bolle magnetiche invece consentono la permanenza, in quanto richiedono unicamente la presenza di un campo magnetico, realizzabile con un magnete permanente. Il problema della permanenza presenta due aspetti:

1) Immagazzinamento dati e programmi a tempo indeterminato; questo può essere assicurato sugli archivi (dischi, nastri). La memoria a bolle fornisce comunque questo tipo di prestazione; il supporto ovviamente non è asportabile.
2) Salvataggio dei risultati dell'ultimo ciclo di operazioni effettuato prima della caduta della tensione di rete: per questo occorre un trasferimento da RAM a memoria ausiliaria non volatile prima che cadano le tensioni continue.

La memoria a bolle si adatta bene a questo tipo di prestazione, grazie anche al suo buon tempo di accesso, in particolare nettamente inferiore ai dischi magnetici; il salvataggio con dischi magnetici è difficoltoso perché occorre tenere le tensioni continue per un tempo troppo lungo.

3.12. MEMORIE CON SUPERFICI MAGNETICHE.

3.12.1. Generalità.

Tra i tipi di memorie a grande capacità, quelli che hanno avuto più larga diffusione sono certamente i **sistemi di memoria con superfici ricoperte di un sottile strato di materiale magnetico**. L'informazione può esservi immagazzinata e letta disponendo le sonde magnetiche di fronte alla superficie, sia per magnetizzare localmente piccole areole, sia per rilevarne lo stato di magnetizzazione.
Per disporre di memorie con tempo di accesso ragionevole, **la forma più comune data a questi sistemi è quella di un disco magnetico**, la cui superficie viene ricoperta con una vernice a base di ossidi metallici. Quando invece si vuole disporre di una **memoria ausiliaria**, usabile anche come unità di ingresso e unità di uscita dei dati, **la forma più comune è il nastro magnetico**.

Le tecnologie di registrazione magnetica possono anche essere usate per realizzare unità di piccole dimensioni e di prestazioni e costo ridotti, dette appunto miniunità di registrazione. Si possono, allo scopo, usare supporti sia a nastro che a disco.

La memorizzazione si compie inviando un impulso di corrente nella sonda di scrittura, in modo da generare un flusso che concatena la areola della superficie magnetica immediatamente di fronte alla sonda. La lettura viene eseguita facendo muovere la superficie di fronte alla sonda di lettura (playback) in modo che, al presentarsi di un'areola magnetizzata, il cambiamento di flusso induca nella bobina della sonda una tensione che costituisce il segnale di uscita.
Due metodi, tecnicamente differenti, vengono in genere adottati per memorizzare le due cifre 0 e 1.
Nel **metodo**, noto con il nome di **registrazione con ritorno a zero (RZ-Return-to-Zero)**, l'areola è magnetizzata in una direzione per rappresentare 1,

e magnetizzata nella direzione opposta per rappresentare 0 (fig. 3.33 (a)). Nel **secondo metodo**, detto di **registrazione senza ritorno a zero (NRZ-Non-Return-to-Zero)**, si ha un cambiamento della direzione di magnetizzazione solo quando si deve scrivere 1 (fig. 3.33 (b)).
Una variazione del metodo NRZ è riportata in fig. 3.33 (c) in cui la direzione di magnetizzazione cambia solo quando la sequenza delle informazioni passa da 0 a 1 oppure da 1 a 0.
Gli impulsi che si prelevano sulla sonda di lettura a seconda dei vari metodi usati sono riportati in fig. 3.33 a', b', c'.
Poiché con il metodo NRZ la direzione di magnetizzazione cambia meno frequentemente che con il metodo RZ, si raggiunge una utilizzazione più efficiente del mezzo magnetico.
E' evidente infatti che con il metodo NRZ le testine di sonda e i circuiti associati possono rispondere ad una frequenza massima di successione di bit che, in prima approssimazione, è il doppio di quella accettabile con il metodo RZ. Ciò determina anche il numero di bit che possono essere contenuti su una traccia per unità di lunghezza.

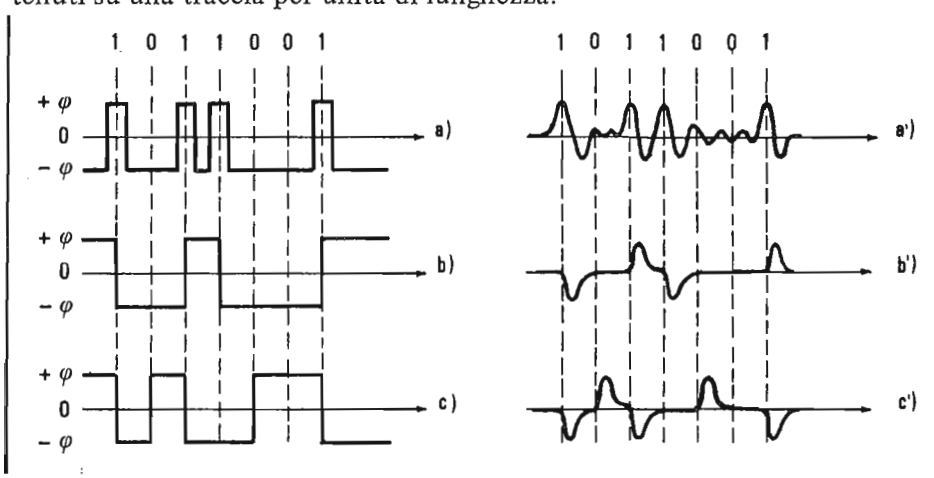


Fig. 3.33 - Tipi di segnali per la scrittura e la lettura su superfici magnetiche.

3.12.2. I dischi magnetici.

I dischi magnetici vengono utilizzati frequentemente nei calcolatori come organi di memoria intermedia a grande capacità. Si distinguono due tipi di unità a disco: quelle a testine fisse e quelle a testine mobili.

Le unità a disco a testine fisse presentano una testina per traccia, essendo le testine posizionate in modo rigido rispetto al disco.
Le unità a disco a testine mobili presentano delle testine in grado di accedere alle varie tracce seguendo un percorso radiale rispetto alla superficie utile del disco.
Nelle unità di registrazione vengono utilizzate per la memorazziazione ambedue le superfici del disco. Oltre che un solo disco, può venire utilizzato un pacco di dischi. Nell'ambito di un pacco di dischi si accede ad un dato disco selezionando elettronicamente la rispettiva testina e spostandola radialmente per accedere alle varie tracce (fig. 3.34).
Il tempo di accesso nelle unità a testina fissa è inferiore a quello delle unità a testina mobile, perché nel caso delle testine mobili occorre aggiungere al tempo medio di accesso sul disco, corrispondente a quello occorrente per una mezza rotazione del disco, il tempo per posizionare il braccio che porta la testina. Questo tempo è di circa un decimo di secondo, superiore quindi di un ordine di grandezza al tempo di rotazione del disco (velocità di rotazione di 60 giri al secondo).
I dati possono essere immagazzinati in diversi modi: ad esempio ogni parola composta di n bit può essere registrata inviando gli n bit in parallelo su n testine e si ha il metodo bit-parallel digit-serial; mentre, se sulle n testine si inviano in parallelo n parole diverse fornendo in sequenza temporale i bit di ogni parola, si ha il metodo bit-serial digit parallel.

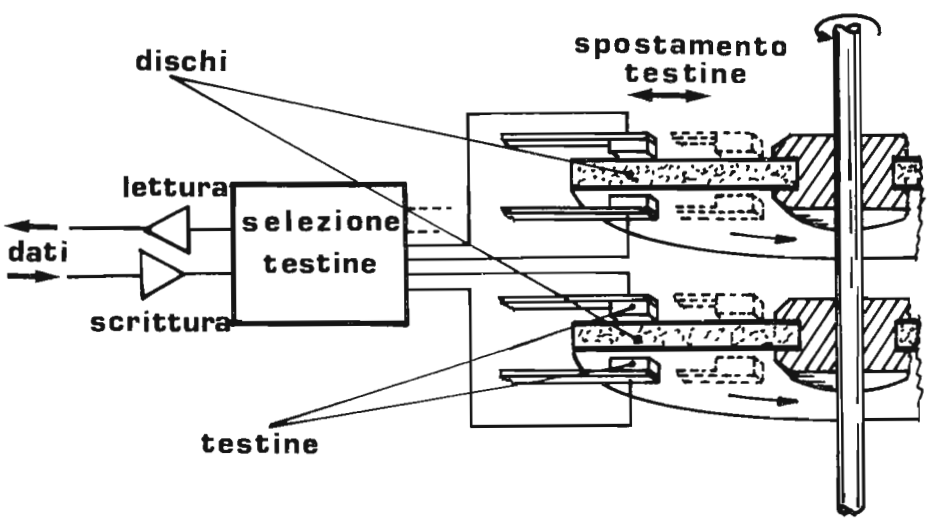


Fig. 3.34 - Sezione di unità a pacco di dischi magnetici a testine mobili.

Le informazioni sul disco vengono organizzate secondo lo schema di fig. 3.35.
Oltre alle tracce destinate alla memorizzazione dei dati, che possono arrivare fino a diverse centinaia , esistono le piste di servizio quali la pista "clock di origine" che dà un segnale ad ogni giro del disco, la pista "clock di settore" che indica l'inizio di un blocco di dati e la pista clock di bit che fornisce la posizione delle celle individuali di memoria.

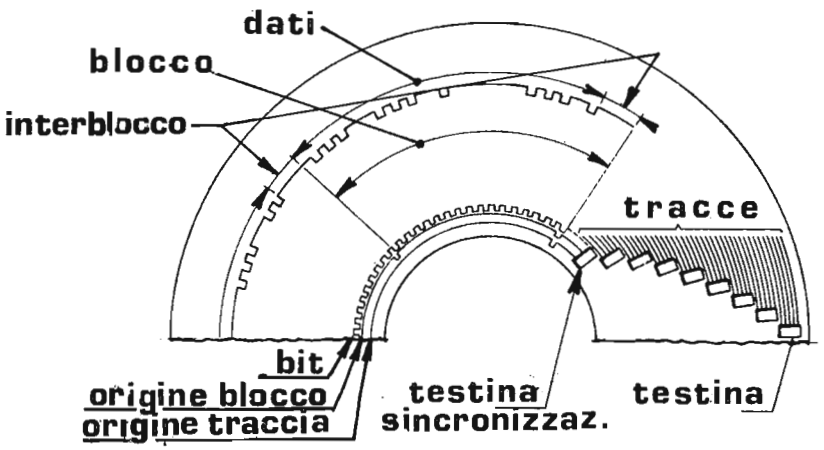


Fig. 3.35 - Disco magnetico con organizzazione dei dati.

3.12.3. I nastri magnetici .

Il nastro magnetico ha la sua applicazione più importante come mezzo per l'ingresso e l'uscita dei dati ad alta velocità. Viene anche utilizzato come memoria ausiliaria, ma in questo caso la limitazione più grave è il lungo tempo di accesso.
Il tipo di nastro più usato è formato da una striscia flessibile di materiale plastico (il mylar è in genere preferito per la resistenza meccanica allo strappo) spesso circa 0,01 pollice e largo da 0,25 a 2 pollici, su cui si deposita una vernice di materiale magnetico come l'ossido di ferro.

In questo caso **la capacità di memoria è in relazione all'area del nastro.** Tuttavia un fattore altrettanto determinante è il modo in cui il nastro viene

usato: esso può scorrere con continuità davanti alle testine di sonda, oppure ad intermittenza con comando di partenza ed arresto fornito dal calcolatore. In questo secondo caso è inevitabile che vada perduta ai fini della registrazione la porzione di nastro che scorre sotto le testine durante i tempi di accelerazione e di frenata.

Per fissare gli ordini di grandezza, si deve tener presente che il nastro può scorrere a velocità che variano da 10 a 150 pollici al secondo. Quest'ultima velocità equivale a circa 1/8 di pollice per millisecondo, e poichè si riesce a raggiungere dei tempi di arresto e di partenza di 5-6 millisecondi, si vede che è necessario lasciare circa 2 pollici di nastro fra i successivi blocchi di informazione: è ovvio allora che per raggiungere una più elevata capacità di memoria è vantaggioso fare più lunghi possibili i blocchi di dati da registrare fra un'operazione di partenza e di arresto.

E' interessante il meccanismo di trasporto del nastro che permette di raggiungere questi tempi di arresto e partenza. Fra le bobine e le testine di registrazione si fa in modo che rimanga pendente una porzione di nastro (fig. 3.36). Poiché l'inerzia del nastro libero in caduta è molto minore di quella delle bobine avvolgitrici, si riesce ad accelerare e decelerare rapidamente il nastro con un sistema formato da due argani, uno in rotazione e l'altro fermo. e una puleggia a folle su cui il nastro è posato. Lo spostamen-

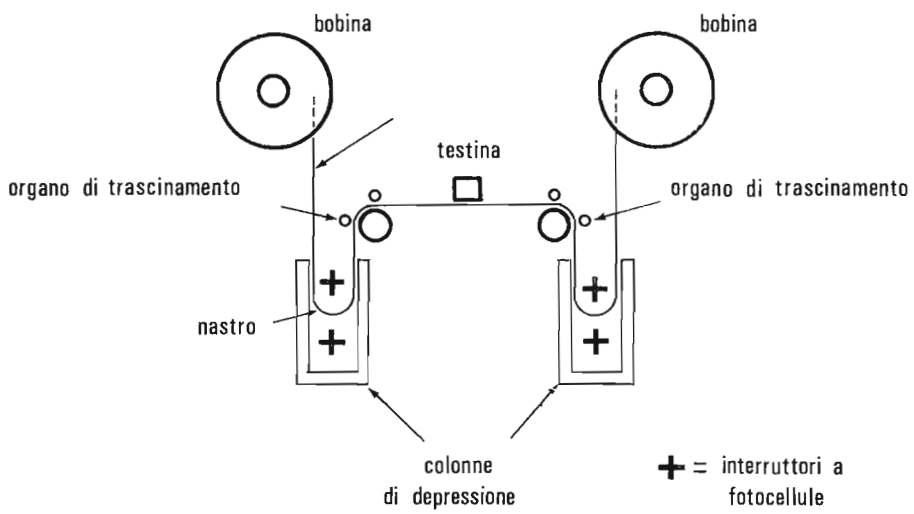


Fig. 3.36 - Schema di principio per il trasporto del nastro magnetico.

to o l'arresto del nastro si ottiene per frizione fra un argano e la puleggia. Giacchè l'avvolgimento deve poter avvenire in entrambi i versi, è necessario disporre di due di questi sistemi perchè il nastro deve esser tirato e non spinto. Le bobine seguono il moto con un leggero ritardo. Le porzioni di nastro in caduta sono inserite in due colonnine in cui si pratica un leggero vuoto, sicchè la pressione esterna le mantiene tese. Per prevenire che dette porzioni divengano troppo corte o troppo lunghe in relazione al tempo di intervento delle bobine, due fotocellule segnalano e controllano il livello massimo ed il livello minimo consentito alla spira del nastro in ogni colonnina.
Normalmente un nastro completo ha una lunghezza di 2400 piedi, eccezionalmente può raggiungere anche 3600 piedi. La velocità di avvolgimento può variare da 10 a 100 pollici/sec., sebbene ci si orienti verso valori standard di 36 e 80 pollici/sec.. La densitá di caratteri per pollice può variare da 200 a 1200 con valori standard intermedi di 556 e 800 caratteri per pollice. Si vede così che un **nastro magnetico ha una capacità di memoria che può variare da 10^7 a 10^8 bit.**

3.12.4. Le mini unità di registrazione.

Con l'avvento dei piccoli sistemi di elaborazione e delle apparecchiature per la raccolta e trasmissione dei dati, le tecniche di registrazione magnetica vengono pure impiegate nella fabbricazione di supporti sia a nastro che a disco, aventi prestazioni e costo ridotti rispetto a quelli delle memorie magnetiche di massa vere e proprie, caratterizzate da elevate capacità e prestazioni come descritto nei paragrafi precedenti.
Le unità di registrazione più diffuse di queste classe sono le "cassette" e i "dischetti".

Il supporto dati trattato dalle unità cassette e rappresentato da una cassetta di nastro di tipo professionale derivata dalla cassetta musicale Philips (si veda fig. 3.37) con alcune differenze fondamentali:

1) Nastro speciale, per impieghi digitali, certificato, cioè garantito all'origine, come privo di "drop-out" (buchi o cadute di segnale) superiori al 10% o "drop-in" (aumenti di segnale) superiori al 25% rispetto alla

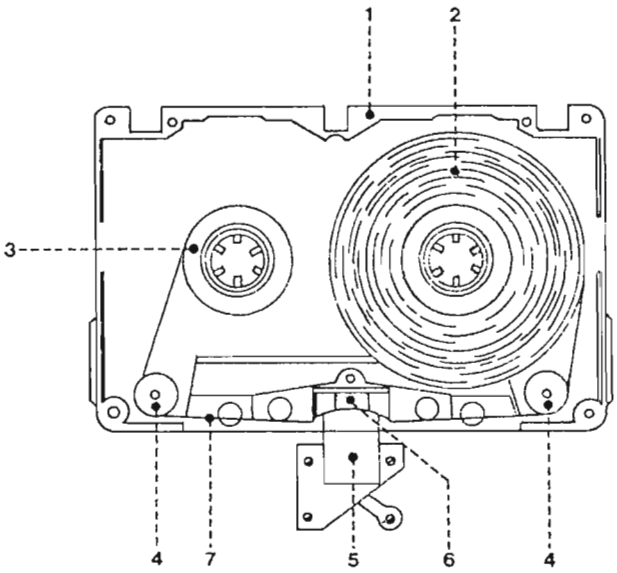


Fig. 3.37 – Cassetta magnetica.
1. Struttura cassetta - 2. Bobina completamente avvolta - 3. Mozzò bobina - 4. Rullini di guida del nastro - 5. Testina magnetica (sulla macchina) - 6. Pattino pressore - 7. Percorso del nastro.

ampiezza standard, misurata alla prima passata dopo la registrazione. Rappresenta una condizione importante per il conseguimento di un basso error rate e di un elevato numero di passate (circa 1.000).
E' simile ai valori di certificazione delle unità nastro, delle quali è praticamente identico il supporto magnetico.

2) Attrito bobine: viene controllato e garantito inferiori a un certo livello; è essenziale per il funzionamento della macchina. Infatti il nastro è recuperato da un motore applicante una coppia alla bobina che avvolge. E' importante che questa coppia sia sempre superiore alla coppia massima di attrito, con un certo margine, per garantire il corretto avvolgimento del nastro sulla bobina.
Un attrito contenuto è inoltre importante ai fini di un basso jitter nelle prestazioni dell'unità.

3) Guida del nastro entro la cassetta; operata da rullini spondati di precisione, congiuntamente a piani di riferimento della cassetta, con ristrette tolleranze meccaniche; condizione indispensabile per il funzionamento corretto della macchina che impiega una sola guida di testina semplice e poco costosa.

4) Pattino pressore del nastro contro la testina, più avvolgente ed a pressione uniforme. Se si tiene presente che: a) la testina, di tipo dual gap, che cioè non ha i gaps (scrittura e lettura) alla sommità, ed è, quindi, a raggio costante; b) lo spessore dell'ossido sul nastro è di soli 5 μm, per ottenere un discreto impaccamento, si capirà come queste due circostanze determinino una esaltazione dell'effetto dei fenomeni transitori di allontanamento del nastro dalla testina: la prima perché, data la forma della testina, è più facile che il fenomeno si manifesti; la seconda perché un distacco dalla testina, dello stesso ordine di grandezza dello spessore dell'ossido, è sufficiente per produrre elevati effetti di drop-out, e come, di conseguenza, il pattino speciale sia importante per il conseguimento di un basso tasso di errore.

Il supporto dati trattato dalle unità a minidisco o dischetti è costituito da un dischetto flessibile ("floppy disk"), simile in apparenza agli analoghi dischi fonografici.
Il supporto magnetico (fig. 3.38) è costituito da un disco di Mylar dello spessore di 0,075 mm, sulle due superfici del quale è depositato uno strato di ossido magnetico.

Questo strato è particolarmente sottile (3μm) ed ha tuttavia elevate caratteristiche di resistenza grazie all'inserimento di opportuni lubrificanti solidi nella miscela che costituisce l'ossido magnetico.

La testina ha un gap di lettura e registrazione più uno di cancellazione; il contatto testina-disco può essere ridotto ad un'area piccolissima.

Questo fatto aiuta esso pure ad aumentare la vita del supporto e delle testine che di per se stesse sono fatte con i migliori materiali antiusura derivati dall'esperienza su nastri.

Il contatto stabile testina-disco e la velocità costante hanno permesso di elevare l'impaccamento oltre i 3.200 bpi (NRZ 2F). Si può notare che questo è il più alto impaccamento raggiunto con tale tecnica di registrazione su macchine commerciali. Il contatto è ottenuto con un pressore che esercita una debole pressione sul disco spingendolo verso la testina.

Il supporto è contenuto in un caricatore costituito da una grossa busta di materiale plastico la quale ha un foro al centro per permettere al mandrino del trascinatore di ingaggiare il disco, ed un foro radiale ad isola nel quale la testina è posta in contatto con il disco stesso.

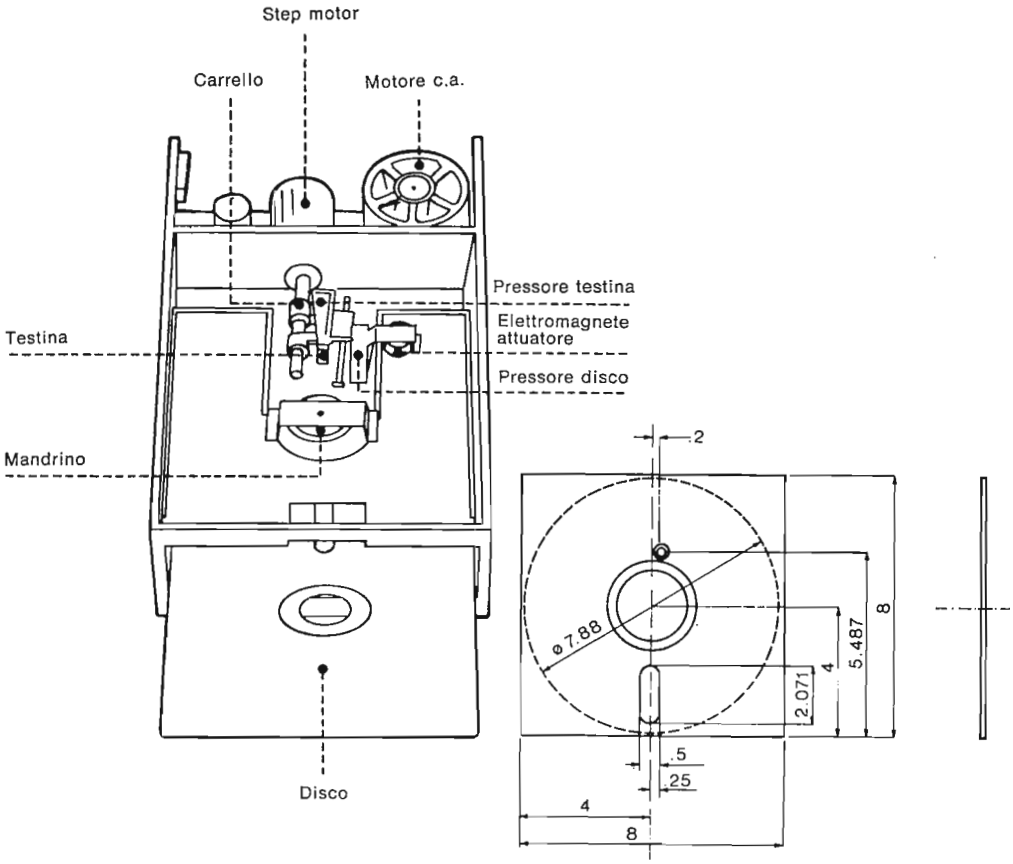


Fig. 3.38 – Unità floppy disk - Cartridge (le dimensioni sono in pollici).

Le pareti interne del caricatore sono rivestite di un tessuto di tipo cellulosico impregnato di sostanza a base siliconica per ridurre al minimo l'attrito disco e al tempo stesso trattenere nei suoi alveoli le sostanze estranee che si depositano sul disco, oltre all'ossido magnetico che viene distaccato dalla testina.

Per rendere questa azione più efficace, nelle fasi di lettura e registrazione, il caricatore viene compresso leggermente contro il disco, nell'area che precede immediatamente la posizione della testina. La somma di questi accorgimenti e soluzioni tecniche, ha dato, come risultato, valori di durata eccezionale.

Alla velocità di 6 giri/sec.un disco tipico ha una vita di almeno 10^6 giri per ogni posizione di pista, con testina in contatto. La vita, con testina non a contatto, è dell'ordine di due anni di rotazione continua.

I dischetti presentano un tempo medio di accesso di parecchio inferiore al secondo ed una velocità di trasferimento di qualche centinaio di migliaio di bit/sec, mentre le cassette presentano un tempo medio di accesso dell'ordine delle decine di secondi ed una velocità di trasferimento di qualche decina di migliaia di bit/sec.
Le capacità di memorizzazione per supporto sono invece comparabili (fino a circa un milione di bit).

Capitolo 4

STRUTTURE SOFTWARE.

4.1. GENERALITA' SULLE TECNICHE DI PROGRAMMAZIONE.

La superiorità dei minicomputers rispetto agli altri strumenti elettronici, sia nella concezione strutturale che nelle prestazioni applicative, va ricercata nel fatto che in essi le operazioni da eseguire non sono prestabilite e limitate dallo schema circuitale adottato, ma possono essere **programmate** a seconda dell'applicazione desiderata: il programma essendo di volta in volta immagazzinabile in memoria.
Per questo nei sistemi di strumentazione elettronica si sogliono distinguere le apparecchiature di acquisizione ed elaborazione di dati in due classi: la prima comprende i cosiddetti **"strumenti a programma cablato"**, cioè che hanno la possibilità di eseguire uno o più programmi già predeterminati dallo schema circuitale, non modificabili ulteriormente e predisponibili con comando di commutatori ed interruttori; la seconda comprende invece gli **"strumenti a programma memorizzato"** che hanno quindi la possibilità di essere programmati per numerose e diverse applicazioni semplicemente inserendo in memoria un diverso programma formulato e costruito con una sequenza di istruzioni scelte fra quelle proprie dello strumento usato.
Il minicomputer è lo strumento più significativo di questa seconda classe grazie al fatto che proprio la versatilità di applicazione ne ha favorito la produzione in quantità di serie ed ha perciò consentito di raggiungere prezzi unitari molto inferiori agli strumenti "a programma cablato" dedicati ad una specifica applicazione.

E' evidente che per scrivere il programma occorre scegliere l'**algoritmo** di soluzione, cioè definire la sequenza o la lista di istruzioni per ottenere l'esecuzione del problema applicativo a singoli passi successivi.
Un esempio molto bello per illustrare e definire cosa è un algoritmo, è quello della coreografia di un balletto: la danza è decomposta in posizioni e passi singoli che messi insieme in sequenze diverse consentono di ottenere una indefinita varietà di balletti.
Nello stesso modo un algoritmo eseguito da un calcolatore può combinare operazioni aritmetiche e logiche in milioni di passi successivi con sequenze diverse ed eseguire complicati problemi.
L'algoritmo per risolvere un problema, ad esempio per il calcolo della funzione $y = f(x)$, **esiste se si può indicare una sequenza ordinata di passi discreti**, eseguibili da una macchina, in modo tale che con un numero finito di detti passi la macchina può calcolare $y = f(x)$ ed eseguire il problema, oppure indicare che non è possibile e non esiste la soluzione.
Per sviluppare la soluzione di un problema con un calcolatore occorre perciò procedere ordinatamente alle seguenti fasi di lavoro:

I^ fase – **Definizione ed analisi del problema**: la definizione del problema non sempre è una cosa ovvia e spesso richiede un grande impegno di lavoro e di tempo. Infatti se è semplice definire il problema quando ad esempio si tratta di eseguire la somma di una lista di numeri, non lo è altrettanto quando si deve individuare e descrivere un problema di controllo di processo. Anche in un caso abbastanza semplice come quello ad esempio del collaudo preventivo di un aereo prima di una missione in volo, molti sono gli interrogativi del tipo: "cosa deve essere soddisfatto del programma di prova?". Si deve allora non solo definire il problema ma anche eseguirne l'analisi individuando gli interrogativi che devono essere risolti e descrivendo i risultati da raggiungere.

II^ fase – **Determinazione ed analisi del metodo di soluzione**: vi è sempre più di un metodo per risolvere un problema. Occorre allora proporre una soluzione e definire un algoritmo di soluzione scelto fra quelli che sono più adatti al problema stesso.

III^ fase – **Codificazione della soluzione nel linguaggio di programmazione**: definito l'algoritmo di soluzione del problema, occorre poi tradurlo nei termini più adatti al tipo di calcolatore adoperato. Spesso la scelta di un metodo di soluzione rispetto all'altro è pro-

prio influenzata dal sistema di calcolo disponibile. E' questa la fase in cui si elaborano i diagrammi di flusso in forma grafica per rappresentare i passi logici della soluzione, e si procede quindi alla codificazione della soluzione nel linguaggio di programmazione del calcolatore. Quando il programma è stato codificato e le istruzioni del programma sono memorizzate nel calcolatore si può procedere alla soluzione del problema.

IV^ fase – **Controllo della codificazione del programma**: non è frequente il caso che un programma funzioni subito senza errori come ci si aspetterebbe dopo la fase di una corretta decodificazione. E' opportuno perciò procedere dopo la stesura del programma ad un controllo o "debugging" del programma stesso. Questa fase richiede perciò di controllare metodicamente, con un procedimento passo passo, il flusso delle istruzioni al fine di trovare gli errori che possono esistere.
Sono stati sviluppati approcci e tecniche diverse per controllare la correttezza di un programma e per elaborare programmi automatici di "debugging".

V^ fase – **Esecuzione automatica del programma di soluzione**: è questa la fase in cui interviene esclusivamente il calcolatore a risolvere in modo automatico il problema secondo il programma codificato in memoria.

VI^ fase – **Controllo ed analisi dei risultati dell'elaborazione**: dopo l'esecuzione dell'elaborazione segue un'ulteriore fase di controllo e di analisi dei risultati conseguiti. Se l'analisi rivela ulteriori errori, si procede alla loro correzione risalendo, per gradi successivi, al controllo del lavoro svolto nelle fasi precedenti: prima si verifica se sono stati commessi errori del calcolatore; quindi se permangono errori di traduzione nella codificazione del programma; quindi ancora se l'algoritmo proposto per la soluzione contenga errori logici; ed infine se sia errata qualche parte nell'analisi o nella definizione del problema.

Come si vede tutta l'applicazione prevede molte altre fasi di lavoro oltre a quella, la quinta nella nostra esposizione, del puro calcolo meccanico o elaborazione elettronica automatica. Ciò presuppone quindi che, per raggiungere la multiforme e versatile capacità di applicazione dello strumento a programma memorizzato, **cioè del calcolatore**, siano svolte da parte **dello**

utilizzatore, cioè dell'uomo, tutte le altre fasi che in sintesi consistono di:

1 - definizione e analisi del problema e del metodo di soluzione;
2 - programmazione dell'algoritmo di soluzione nel linguaggio del calcolatore;
3 - analisi e controllo dei risultati;

operazioni queste che negli strumenti a programma cablato sono state risolte dal progettista dello strumento e che perciò possono venire più semplicemente predisposte con comandi manuali di commutatori ed interruttori.
E' per questo motivo che, mentre l'uso di uno strumento a programma cablato è quasi immediato, viceversa l'uso efficiente di un calcolatore richiede tutto un lavoro supplementare che per essere svolto validamente comporta tutto un corredo di mezzi logici e di programmazione che va sotto il nome di "software".
Tutto il corredo di software di un calcolatore può essere ben suddiviso e catalogato secondo questa classificazione:

– **SOFTWARE DI BASE**: comprende i linguaggi di programmazione del tipo assemblatore, compilatore ed interprete con i relativi programmi di traduzione nel linguaggio macchina. Normalmente oltre all'**assemblatore**, i minicomputers sono corredati anche con i compilatori FORTRAN, ed i programmi interpreti conversazionali BASIC.

– **SOFTWARE DI SERVIZIO** (Utilities): comprendono i programmi, associati a quelli di traduzione del tipo assemblatore o compilatore, che aiutano a semplificare, a controllare a provare la scrittura dei programmi applicativi.
Essi sono noti con il nome di programmi **"Editor"**, **"Loader"**, **"Linking"**, **"Debugging"**, e così via.

– **SOFTWARE DI SISTEMA** (Operating Systems): comprende i cosiddetti "programmi operativi" destinati a gestire, regolare e controllare il funzionamento di un sistema di strumentazione in cui l'unità centrale del calcolatore è integrata con altre unità periferiche quali memorie a disco e a nastro magnetico, stampanti, visualizzatori e altre unità di ingresso dati quali convertitori analogico-numerici e similari.
Questi programmi, detti anche "**sistemi operativi**", gestiscono e ripartiscono il lavoro delle varie unità utilizzando le risorse del sistema integrato, quali capacità di memoria, tempi di elaborazione, tempi di

ingresso-uscita dati, in funzione delle richieste del programma applicativo e delle possibilità proprie alle unità del sistema ed al sistema nel suo insieme.
Ad esempio i minicomputers vengono spesso corredati dei sistemi DOS (Disc-Operating-System) e MTOS (Magnetic-Tape-Operating-System).
Questi sistemi operativi sono di grande vantaggio nel semplificare i programmi applicativi e liberarli dal compito di organizzare e gestire i compiti delle varie unità componenti un sistema standard di elaborazione.

– **SOFTWARE APPLICATIVO** (Application Packages): comprende programmi propri di una determinata applicazione e per questo destinati ad essere sviluppati dall'utilizzatore del minicomputer o su sua commissione. Spesso lo stesso costruttore può fornire alcuni programmi applicativi quando essi si dimostrino di una validità abbastanza generalizzabile quali ad esempio: programmi di "linearizzazione", "conversione di scala", "integrazione" di dati nelle applicazioni per acquisizione di dati; oppure programmi di "normalizzazione", "correlazione", "integrazione di picchi", "filtraggio" nell'analisi di spettri di misura.

4.2. LISTA DELLE ISTRUZIONI TIPICHE DI UN MINICOMPUTER.

Tutto il corredo software di un calcolatore è basato sull'insieme delle istruzioni di macchina, cioè sui comandi codificati che permettono ai circuiti della CPU di compiere le operazioni richieste.
Già nei paragrafi precedenti abbiamo esaminato la struttura cioè il formato in codice binario con cui si presentano le istruzioni e abbiamo visto che le istruzioni si possono raggruppare nelle tre classi generali:

– istruzioni di riferimento alla memoria;
– istruzioni di operazione dei registri;
– istruzioni di ingresso-uscita.

In questo paragrafo vogliamo descrivere i tipi di istruzione più comuni alla maggioranza dei minicomputers e che definiscono le principali operazioni logiche ed aritmetiche svolte.

Da modello a modello il numero complessivo delle istruzioni può variare in funzione sia della versatilità e flessibilità della struttura della CPU, sia di particolari funzioni introdotte per facilitare particolari tipi di applicazione

4.2.1. Istruzioni di riferimento alla memoria.

Queste istruzioni specificano le operazioni che devono essere eseguite su di un operando, memorizzato o da memorizzare in una locazione di memoria. Come abbiamo già visto esse specificano nel campo operativo, comprendente ad esempio i bit 15 ÷ 11, il tipo di operazione mentre nel campo esecutivo dei bit restanti 10 ÷ 0 specificano l'indirizzo della locazione e le modalità per la ricerca dell'indirizzo in cui l'operando è contenuto.
A loro volta queste istruzioni possono essere suddivise, per il significato delle operazioni che fanno svolgere, nei seguenti sottogruppi:

- istruzioni di trasferimento fra memoria e registri;
- istruzioni aritmetiche;
- istruzioni logiche;
- istruzioni di controllo.

Nel seguito definiamo prima il significato delle istruzioni base di ogni sottogruppo e poi specifichiamo l'elenco delle istruzioni con il codice operativo, il nome simbolico, e le azioni da esse determinate riferendoci ad uno specifico modello di minicomputer come il LABEN 70.

Nel sottogruppo delle istruzioni di trasferimento sono comprese essenzialmente le seguenti due istruzioni:

"Load" – determina il trasferimento del contenuto di una specificata locazione di memoria in un registro: caricamento di registro.

"Store" – determina il trasferimento del contenuto di un registro in una specificata locazione di memoria: immagazzinamento in memoria.

Nel sottogruppo delle istruzioni aritmetiche sono comprese sempre le seguenti:

"Add" – il contenuto di una specificata locazione di memoria è addizionato a quello di un registro;

"Subtract" – il contenuto di una specificata locazione di memoria è sottratto a quello di un registro;

"Multiply" – il contenuto di un registro accumulatore viene moltiplicato per il contenuto di una specificata locazione di memoria ed il risultato mantenuto in un registro a lunghezza doppia formato ad esempio da due registri accumulatori;

"Divide" – il contenuto di un registro a lunghezza doppia è diviso per il contenuto di una specificata locazione di memoria..

Nel sottogruppo delle istruzioni logiche sono comprese sempre le seguenti:

"AND" – un'operazione logica di AND è eseguita fra il contenuto del registro accumulatore e il contenuto di una specificata locazione di memoria.

"OR" – un'operazione logica di OR è eseguita fra il contenuto del registro accumulatore ed il contenuto di una specificata locazione di memoria.

"Exclusive OR" – un'operazione logica di OR-esclusivo è svolta fra il contenuto dell'accumulatore e quello di una specificata locazione di memoria.

Nel sottogruppo delle istruzioni di controllo sono in genere comprese almeno le seguenti:

"JUMP" – il contatore di istruzione viene sostituito da una specificata locazione di memoria: la istruzione successiva viene perciò presa dalla locazione specificata.

"Jump to Subroutine" – il contenuto del contatore di istruzione è trasferito in una specifica locazione di memoria e lo stesso contatore è rimpiazzato dalla locazione specificata più uno.

"Increment"– si aggiunge 1 al contenuto di una specificata locazione di memoria.

"Decrement" – si sottrae 1 dal contenuto di una specificata locazione di memoria.

"Increment (o Decrement) and Skip if Zero" – si aggiunge (o si sottrae) 1 al contenuto di una specificata locazione di memoria e l'istruzione successiva è saltata se il risultato è zero.

4.2.2. Istruzione di operazione dei registri.

La seconda classe di istruzioni concerne **le istruzioni di operazione fra registri**. Esse contengono sempre almeno le seguenti:

"Shift o Spostamento aritmetico" – il contenuto di un registro, ad eccezione del bit di segno, è spostato a destra o a sinistra di un numero specificato di caselle. I bits spostati che vanno fuori le caselle del registro sono perduti.

"Shift o Spostamento logico" – il contenuto di un registro è spostato tutto a destra o a sinistra di un numero specificato di caselle. I bits che vanno fuori con lo spostamento, sono perduti e le caselle lasciate libere dall'altro capo sono messe a zero.

"Rotazione" – il contenuto di un registro viene spostato a destra o a sinistra di un numero di passi specificato. I bits che escono da un capo del registro vengono reintrodotti dall'altro capo procedendo nello stesso verso.

A queste istruzioni fondamentali se ne possono aggiungere altre diverse che facilitano ed estendono le possibilità di programmazione.

4.2.3. Istruzioni di ingresso - uscita.

In questa terza classe di istruzioni sono comprese quelle per la gestione del flusso delle informazioni fra la CPU e le unità di ingresso-uscita. Possono assumere configurazioni e numero diverso da un modello all'altro di calcolatore, ma in generale nella lista si ritrovano almeno le seguenti che costituiscono quelle principali:

"Input" – il contenuto di un registro di memoria temporanea di una unità d'interfaccia è trasferito nell'accumulatore o in una specificata locazione di memoria. Tale contenuto può esprimere un dato o un comando a seconda del tipo di registro della unità d'interfaccia da cui viene trasferito.

"Output" – il contenuto dell'accumulatore o di una specificata locazione di memoria viene trasferito in uno specificato registro di memoria temporanea di una specificata unità periferica.

"Enable/Disable Interrupt" – abilitazione o disabilitazione del sistema di interruzione nel BUS di Input-Output.

"Direct Memory Access" – abilitazione ad un canale periferico per funzionare in accesso diretto alla memoria.

"Set o Reset bistabile Flag o bistabile di controllo" – uno di questi bistabili viene messo ad 1 o viene azzerato in modo da abilitare o meno l'unità periferica al trasferimento dei dati.

"Skip on Flag/Skip on Control" – controlla se il bistabile Flag o il bistabile di controllo dell'unità d'interfaccia selezionata è uguale a 1. In questo caso non viene eseguita l'istruzione successiva di programma, altrimenti si procede secondo la sequenza normale.

"Halt" – viene fermata la sequenza di elaborazione del calcolatore.

Riferendosi al caso di un modello specifico di calcolatore come il LABEN 70, la lista delle istruzioni relative ai sottogruppi ora elencati, la loro codificazione binaria e simbolica, e le azioni da loro determinate possono essere illustrate dalla descrizione che segue.

Istruzioni di trasferimento.

Queste istruzioni sono impiegate per il trasferimento di una parola fra un registro e la memoria e viceversa.

Codice operativo 15 – 11	Codice simbolico	Significato
01001	LDA	Load – Caricamento registro A
01010	LDB	Load – Caricamento registro B
01011	LDX	Load – Caricamento registro X
00101	DLD	Double-Load – Doppio caricamento
11001	STA	Store – Memorizzazione registro A
11010	STB	Store – Memorizzazione registro B
11011	STX	Store – Memorizzazione registro X
01111	DST	Double-Store – Doppia memorizzazione

Consideriamo l'istruzione: **caricamento registro A (LDA) con il contenuto dell'indirizzo simbolico PIPPO:**

LDA PIPPO

Le azioni determinate da tale istruzione sono le seguenti:

- il contenuto della locazione di indirizzo simbolico PIPPO viene trasferito nel registro A;
- il contenuto del registro A, precedente all'esecuzione dell'istruzione, viene distrutto;
- il contenuto della locazione di indirizzo simbolico PIPPO rimane inalterato.

Se, ad esempio, il contenuto della locazione di indirizzo simbolico PIPPO è 100_8, avremo la seguente situazione:

	C(PIPPO)	C(A)
– prima dell'esecuzione dell'istruzione	000100	?
– dopo l'esecuzione dell'istruzione	000100	000100

L'azione determinata dall'istruzione LDA PIPPO può essere compendiata con la seguente notazione:

C(PIPPO) ⟶ C(A)

che sta a significare: il contenuto della locazione di indirizzo simbolico PIPPO viene trasferito nel registro A.

Analogo significato hanno le istruzioni LDB e LDX riferite ai registri B ed X.

Consideriamo l'altra istruzione: **memorizzazione registro A (STA) nella locazione di indirizzo simbolico PIPPO:**

STA PIPPO

Le azioni determinate da tale istruzione sono:

- il contenuto del registro A viene memorizzato nella locazione di indirizzo simbolico PIPPO;
- il contenuto della locazione di indirizzo simbolico PIPPO, precedente all'esecuzione della istruzione, viene distrutto;
- il contenuto del registro A rimane inalterato.

Se, ad esempio, il contenuto del registro A è 100_8, avremo la seguente situazione:

	C(PIPPO)	C(A)
– prima dell'esecuzione dell'istruzione	?	000100
– dopo l'esecuzione dell'istruzione	000100	000100

Con notazione sintetica: C(A) ⟶ C(PIPPO).

Analogo significato hanno le istruzioni STB e STX.

Consideriamo l'istruzione: **doppio caricamento (DLD) con il contenuto dell'indirizzo simbolico PIPPO:**

DLD PIPPO

Le azioni determinate da tale istruzione sono le seguenti:

- il contenuto della locazione di indirizzo simbolico PIPPO viene trasferito nel registro A;
- il contenuto della locazione di indirizzo simbolico PIPPO+1 viene trasferito nel registro B;
- i contenuti dei registri A e B, precedenti all'esecuzione della istruzione, vengono distrutti;
- i contenuti delle locazioni di indirizzi simbolici PIPPO e PIPPO+1 rimangono inalterati.

Con notazione sintetica: C(PIPPO) ⟶ C(A)
C(PIPPO+1) ⟶ C(B)

Analogo significato si deduce per l'istruzione doppia memorizzazione DST.

Istruzioni aritmetiche.

Tali istruzioni sono impiegate per l'esecuzione di operazioni aritmetiche tra un registro e la memoria; ad operazione avvenuta il risultato si trova nel registro.

Codice operativo (15 – 11)	Codice simbolico	Significato
10000	ADA	Addizione al registro A
01000	ADB	Addizione al registro B
11000	ADX	Addizione al registro X
10100	SBA	Sottrazione dal registro A
01100	SBB	Sottrazione dal registro B
11100	SBX	Sottrazione dal registro X
01101	CPL	Complemento
11101	CLR	Azzeramento
00001	MUL	Moltiplicazione
00100	DIV	Divisione

Consideriamo l'istruzione: **addizione al registro A (ADA) con il contenuto del registro simbolico PIPPO**:

ADA PIPPO

Le azioni determinate da tale istruzione sono le seguenti:

- il contenuto della locazione di indirizzo simbolico PIPPO viene sommato al contenuto del registro;
- il risultato viene lasciato nel registro A;
- il contenuto della locazione di indirizzo simbolico PIPPO rimane inalterato.

Se, ad esempio, il contenuto della locazione di indirizzo simbolico PIPPO è 100_8 ed il contenuto del registro A è 1450_8, si ha la seguente situazione:

	C(PIPPO)	C(A)
– prima dell'esecuzione dell'istruzione	000100	001450
– dopo l'esecuzione dell'istruzione	000100	001550

Con notazione sintetica: C(PIPPO) + C(A) ⟶ C(A)

Analogo significato si può dedurre per tutte le altre istruzioni di addizione e sottrazione.

Si consideri l'istruzione: **complemento (CPL) che è ad un operando:**

CPL PIPPO

L'azione determinata da tale istruzione è la seguente:

– il contenuto della locazione di indirizzo simbolico PIPPO viene complementato bit a bit.

Se, ad esempio, il contenuto della locazione di indirizzo simbolico PIPPO è 24363_8, si ha la seguente situazione:

	C(PIPPO)
– prima dell'esecuzione dell'istruzione	0010100011110011
– dopo l'esecuzione dell'istruzione	1101011100001100

L'operazione può anche essere effettuata su un registro, sfruttando la possibilità di indirizzarlo come locazione.

Si consideri l'istruzione: **azzeramento (CLR) che è ad un operando:**

CLR PIPPO

L'azione determinata da tale istruzione è la seguente:

– il contenuto della locazione di indirizzo simbolico PIPPO viene azzerato.

Con notazione sintetica: $0 \longrightarrow$ C(PIPPO)

L'operazione può anche essere effettuata su un registro, sfruttando la possibilità di indirizzarlo come locazione.

Consideriamo l'istruzione: **moltiplicazione (MUL):**

MUL PIPPO

Le azioni determinate da tale istruzione sono le seguenti:

– il contenuto della locazione di indirizzo simbolico PIPPO viene moltiplicato per il contenuto del registro B;
– il prodotto si trova, **in doppia precisione,** nei registri A e B. Il registro A contiene la parte più significativa del prodotto; il bit 15 del registro A è utilizzato come bit di segno del risultato. Il registro B contiene la parte meno significativa del prodotto; il 15 del registro B è forzato a zero;
– il contenuto della locazione di indirizzo simbolico PIPPO rimane inalterato.

Consideriamo l'istruzione: **divisione (DIV):**

DIV PIPPO

Le azioni determinate da tale istruzione sono le seguenti:

– il contenuto dei registri A e B, dividendo in doppia precisione, viene diviso per il contenuto della locazione di indirizzo simbolico PIPPO. Il registro A contiene la parte più significativa del dividendo;

il bit 15 del registro A è utilizzato come bit di segno del dividendo. Il registro B contiene la parte meno significativa del dividendo: il bit 15 del registro B è forzato a zero;

– dopo l'esecuzione dell'istruzione il registro B contiene il quoziente; il bit 15 del registro B è utilizzato come bit di segno. Il registro A contiene il resto della divisione col segno del dividendo.

Istruzioni logiche.

Queste istruzioni sono impiegate per eseguire le operazioni logiche AND, OR e OR-ESCLUSIVO tra la memoria ed il registro A.
Ad operazione avvenuta, il risultato si trova nel registro A o nella locazione di memoria indirizzata, tale duplice possibilità dipendendo dall'istruzione usata.

Codice operativo (15 – 11)	Codice simbolico	Significato
10001	ANA	And logico con risultato nel registro A
10010	ORA	Or logico con risultato nel registro A
10011	ERA	Eor logico con risultato nel registro A
10101	ANM	And logico con risultato in memoria
10110	ORM	Or logico con risultato in memoria
10111	ERM	Eor logico con risultato in memoria

Consideriamo l'istruzione: **And logico con risultato nel registro A(ANA):**

ANA PIPPO

Le azioni determinate da tale istruzione sono le seguenti:

– viene eseguita, bit a bit, l'operazione di And tra il contenuto della locazione di indirizzo simbolico PIPPO ed il contenuto del registro A. Un 1 appare ogniqualvolta è presente un 1 sia nella locazione sia nel registro A;
– il risultato viene lasciato nel registro A;
– il contenuto della locazione di indirizzo simbolico PIPPO rimane inalterato.

Se, ad esempio, i contenuti della locazione di indirizzo simbolico PIPPO e del registro A sono quelli indicati, si ha la seguente situazione:

prima dell'esecuzione dell'istruzione 0110001010011100 registro A
1110111000111000 locazione PIPPO

dopo l'esecuzione dell'istruzione 0110001000011000 registro A
1110111000111000 locazione PIPPO

Da questo esempio, per analogia, si desume il significato anche delle istruzioni ORA ed ERA.

Analoghe considerazioni possono essere fatte anche per le istruzioni ANM, ORM e ERM: in questo caso il risultato viene lasciato nella locazione indirizzata.
Avremo quindi:

Data l'istruzione: **And logico con risultato in memoria (ANM):**

ANM PIPPO

le azioni da essa determinate sono le seguenti:

- viene eseguita, bit a bit, l'operazione di And tra il contenuto della locazione di indirizzo simbolico PIPPO ed il contenuto del registro A;
- il risultato viene lasciato nella locazione di indirizzo simbolico PIPPO;
- il contenuto del registro A rimane inalterato.

Istruzioni di controllo.

Queste istruzioni permettono di modificare la sequenza di esecuzione delle istruzioni.

Codice operativo (15 – 11)	Codice simbolico	Significato
01100	JMP	Salto
11110	JMS	Salto e memorizzazione del contenuto del registro C
00111	XEC	Esecuzione fuori sequenza
00010	CSE	Paragone con salto condizionato
00011	ISZ	Incremento con salto condizionato
01110	DSZ	Decremento con salto condizionato

Consideriamo l'istruzione: **salto (JMP):**

JMP PIPPO

Le azioni da essa determinate sono le seguenti:

- il controllo viene trasferito alla locazione di indirizzo simbolico PIPPO;
- l'esecuzione riprende poi da tale locazione.

Consideriamo, ad esempio, la seguente situazione:

Locazione	Contenuto
.	.
.	.
.	.
250_8	JMP 300_8
.	.
.	.
.	.
300_8	.
.	.
.	.

L'effetto dell'istruzione: JMP 300_8, è di trasferire il controllo direttamente alla locazione di indirizzo assoluto 300_8 e riprendere l'esecuzione a partire da tale locazione, scavalcando, in tal modo, le locazioni di indirizzo assoluto compreso tra: $251_8 - 277_8$.

Consideriamo l'istruzione: **paragone con salto condizionato (CSE)**:

CSE PIPPO

Le azioni determinate da tale istruzione sono le seguenti:

- il contenuto della locazione di indirizzo simbolico PIPPO viene confrontato con il contenuto del registro A;
- se tali contenuti sono uguali, l'istruzione successiva non viene eseguita (il contenuto del registro C viene incrementato di 2 unità); in caso contrario si procede in sequenza;
- tanto il contenuto della locazione di indirizzo simbolico PIPPO quanto il contenuto del registro A rimangono inalterati.

Consideriamo l'istruzione: **incremento con salto condizionato (ISZ)**:

ISZ PIPPO

Le azioni determinate da tale istruzione sono le seguenti:

- il contenuto della locazione di indirizzo simbolico PIPPO viene incrementato di una unità;
- viene quindi esaminato il risultato dell'incremento. Se, a seguito di tale incremento, il contenuto della locazione di indirizzo simbolico PIPPO diventa uguale a zero, l'istruzione successiva non viene eseguita (il contenuto del registro C viene incrementato di 2 unità); in caso contrario, si procede in sequenza;
- in ogni caso il risultato dell'incremento sostituisce il precedente valore contenuto nella locazione di indirizzo simbolico PIPPO;
- deve essere chiaro che il test sul contenuto della locazione di indirizzo simbolico PIPPO è fatto **dopo** l'incremento.

Consideriamo l'istruzione: **decremento con salto condizionato (DSZ)**:

DSZ PIPPO

Le azioni da essa determinate sono le seguenti:

- il contenuto della locazione di indirizzo simbolico PIPPO viene decrementato di una unità;

- viene quindi esaminato il risultato del decremento. Se, a seguito di tale decremento, il contenuto della locazione di indirizzo simbolico PIPPO diventa uguale a zero, l'istruzione successiva non viene eseguita (il contenuto del registro C viene incrementato di 2 unità); in caso contrario, si procede in sequenza;
- in ogni caso il risultato del decremento sostituisce il precedente valore contenuto nella locazione di indirizzo simbolico PIPPO;
- deve essere chiaro che il test sul contenuto della locazione di indirizzo simbolico PIPPO è fatto **dopo** il decremento.

Le istruzioni ISZ e DSZ vengono usate per la realizzazione di cicli, quando si ha a che fare con calcoli di tipo ripetitivo.

Consideriamo l'istruzione: **esecuzione fuori sequenza (XEC):**

XEC PIPPO

Le azioni da essa determinate sono le seguenti:

- il controllo viene trasferito alla locazione di indirizzo simbolico PIPPO e viene eseguita l'istruzione contenuta in tale locazione: tale istruzione non deve modificare il registro C;
- dopo aver eseguito l'istruzione contenuta nella locazione di indirizzo simbolico PIPPO, il controllo viene trasferito all'istruzione immediatamente seguente l'istruzione XEC.

Consideriamo l'istruzione: **salto e memorizzazione del contenuto del registro C (JMS):**

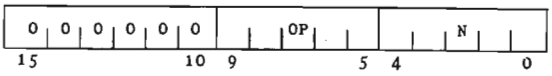
JMS PIPPO

Per comprendere le azioni determinate da tale istruzione è necessario riferirsi al concetto di sottoprogramma o subroutine: quando una serie di istruzioni viene incontrata più volte nel corso del programma, essa può essere codificata a parte come sottoprogramma. Tale possibilità è ottenuta ricorrendo all'istruzione JMS.
Le azioni determinate da tale istruzione sono le seguenti:

- il contenuto del registro C incrementato di una unità viene memorizzato nella locazione indirizzata (prima locazione del sottoprogramma), distruggendo il contenuto precedente;
- il controllo viene trasferito alla seconda locazione del sottoprogramma.

Vengono quindi eseguite le istruzioni costituenti il sottoprogramma. Il sottoprogramma è concluso dall'istruzione: JMP SUBR. Grazie a tale istruzione il controllo viene nuovamente trasferito al programma principale in corrispondenza all'istruzione successiva all'istruzione di JMS.

Nel caso del LABEN 70 il formato dell' istruzioni di spostamento, rotazione, posizionamento dei registri di carry e di overflow si presenta con la codificazione seguente:



Il campo costituito dai bit 15-10 è fisso e caratteristico di questo gruppo di istruzioni.
Il campo OP (bit 9-5) specifica il codice operativo delle singole istruzioni del gruppo.
Il campo N (bit 4-0) nel caso delle istruzioni di spostamento contiene il numero di posizioni di cui deve essere spostato il contenuto del 0 dei registri interessati dall' istruzione; nel caso di istruzioni di posizionamento dei registri di extend e di overflow e di non operazione è sempre uguale a zero.

Nella tabella seguente sono riportati i codici operativi in rappresentazione binaria, i codici simbolici ed il significato di tutte le istruzioni di questo gruppo.

Tabella dei codici delle istruzioni di spostamento, di non operazione, di posizionamento dei registri di Overflow e di Extend

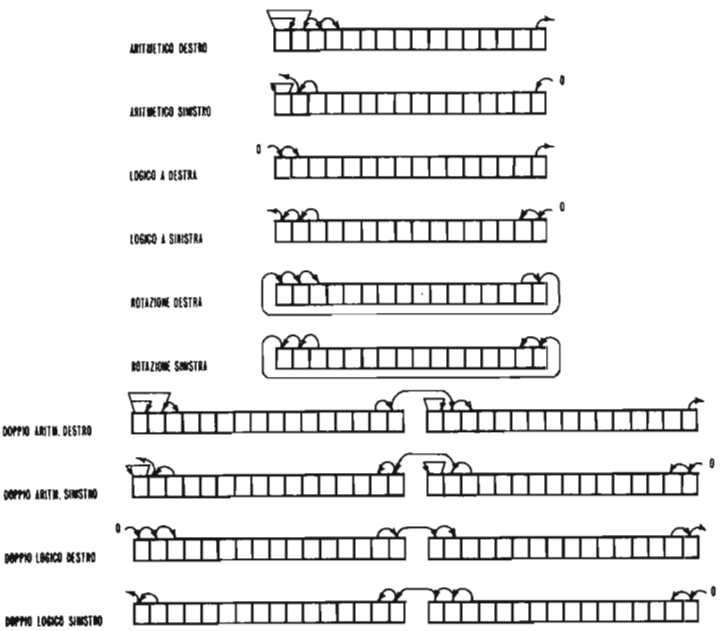
Contenuto campo OP	Codice simbolico	Significato
00011	ARA	Spostamento aritmetico destro del registro A
00010	ALA	Spostamento aritmetico sinistro del registro A
00101	ARB	Spostamento aritmetico destro del registro B
00100	ALB	Spostamento aritmetico sinistro del registro B
00111	ARX	Spostamento aritmetico destro del registro X
00110	ALX	Spostamento aritmetico sinistro del registro X
01011	LRA	Spostamento logico destro del registro A
01010	LLA	Spostamento logico sinistro del registro A
01101	LRB	Spostamento logico destro del registro B
01100	LLB	Spostamento logico sinistro del registro B
01111	LRX	Spostamento logico destro del registro X
01110	LLX	Spostamento logico sinistro del registro X
10011	RRA	Rotazione destra del registro A
10010	RLA	Rotazione sinistra del registro A
10101	RRB	Rotazione destra del registro B
10100	RLB	Rotazione sinistra del registro B
10111	RRX	Rotazione destra del registro X
10110	RLX	Rotazione sinistra del registro X
11001	ADR	Spostamento aritmetico doppio destro dei registri A e B
11000	ADL	Spostamento aritmetico doppio sinistro dei registri A e B
11111	LDR	Spostamento logico doppio destro dei registri A e B
11110	LDL	Spostamento logico doppio sinistro dei registri A e B
01000	STO	Posizionamento del registro di Overflow
11010	STE	Posizionamento del registro di Extend
00000	NOP	Non operazione

Nella Figura seguente è riportata una rappresentazione grafica delle modalità con cui avvengono i vari tipi di spostamento.

E' chiaro che per la corretta esecuzione delle istruzioni di spostamento dovranno essere fornite le seguenti informazioni:

- tipo di spostamento
- registro o registri interessati.

Tali informazioni vengono ottenute tramite il codice operativo. Dal punto di vista della programmazione simbolica è quindi utile esaminare con più attenzione i codice simbolici, usati dal programmatore, in modo da metterli più facilmente in relazione con le operazioni che realizzano.



LABEN 70 – TIPI DI SPOSTAMENTO

I codici simbolici sono costituiti da tre lettere, il cui significato è il seguente:

- prima lettera: precisa il tipo di spostamento. La relazione intercorrente tra la lettera e il tipo di spostamento è mostrata dalla seguente tabella.

lettera	tipo di spostamento
A	aritmetico
L	logico
R	rotazione

– seconda lettera: precisa la direzione in cui avviene lo spostamento. Avremo la seguente tabella di corrispondenza.

lettera	direzione dello spostamento
L	sinistra (left)
R	destra (right)

– terza lettera: precisa il registro interessato dalla istruzione di spostamento. La corrispondenza tra lettera e registro (del tutto spontanea) è mostrata nella seguente tabella.

lettera	registro interessato
A	registro A
B	registro B
X	registro X

Rimangono le istruzioni di spostamento doppio. In tal caso non vi è la necessità di dare l'indicazione del registro, poiché tali istruzioni si applicano ai registri A e B, considerati come un unico registro di 32 bit.
In questo caso il significato delle lettere è il seguente:

– prima lettera: precisa il tipo di spostamento.

lettera	tipo di spostamento
A	aritmetico
L	logico

– seconda lettera: tale lettera è sempre D e specifica spostamento doppio.
– terza lettera: precisa la direzione in cui avviene lo spostamento.

lettera	direzione dello spostamento
L	sinistra
R	destra

Illustriamo, con alcuni esempi, le azioni determinate dalle istruzioni di spostamento.
Consideriamo l'istruzione: **istruzioni di spostamento aritmetico**:

ARB 3

Vengono mostrati i contenuti del registro B prima e dopo l'esecuzione dell'istruzione.

registro B | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | prima

registro B | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | dopo

Le azioni determinate da una istruzione di spostamento aritmetico destro sono le seguenti:

- il contenuto del registro interessato viene spostato a destra di tante posizioni quante sono indicate nel campo variabile;
- il bit 15 del registro interessato resta immutato e viene ripetuto in tutte le posizioni lasciate libere a sinistra.

Consideriamo l'istruzione: **istruzioni di spostamento logico**:

LRX 010

Vengono mostrati i contenuti del registro X prima e dopo l'esecuzione dell'istruzione.

registro X | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | prima

registro X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | dopo

Le azioni determinate da una istruzione di spostamento logico destro sono le seguenti:

- il contenuto del registro interessato viene spostato a destra di tante posizioni quante sono indicate nel campo variabile;
- nelle posizioni lasciate libere a sinistra vengono introdotti degli zeri.

Consideriamo l'istruzione: **istruzioni di rotazione**:

RRB 2

Vengono mostrati i contenuti del registro B prima e dopo l'esecuzione dell'istruzione.

registro B	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	prima
registro B	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	dopo

Le azioni determinate da una istruzione di rotazione destra sono le seguenti:

- il contenuto del registro interessato viene ruotato a destra di tante posizioni quante sono specificate nel campo variabile;
- il bit 0 del registro interessato viene trasferito, durante ogni spostamento, nel bit 15 del registro stesso.

Per illustrare le istruzioni di spostamento aritmetico doppio consideriamo la ADR che causa le seguenti azioni nei registri A e B:

- il contenuto dei registri A e B, considerati come un unico registro, viene spostato a destra di tante posizioni quante sono indicate nel campo variabile;
- il bit 15 dei registri A e B restano immutati;
- il bit 0 del registro A viene trasferito, durante ogni spostamento, nel bit 14 del registro B;
- il bit 15 del registro A viene ripetuto in tutte le posizioni lasciate libere a sinistra.

L'istruzione di spostamento logico doppio destro LDR dei registri A e B causa le seguenti azioni:

- il contenuto dei registri A e B viene spostato a destra di tante posizioni quante sono indicate nel campo variabile;
- il bit 0 del registro A viene trasferito, durante ogni spostamento, nel bit 15 del registro B;
- nelle posizioni lasciate libere a sinistra vengono introdotti degli zeri.

Vi sono poi le istruzioni che consentono di posizionare a 1 i registri di extend e di overflow.
Il codice simbolico della istruzione di posizionamento a 1 del registro di overflow (set-overflow) è STO.
A seguito di tale istruzione il contenuto del registro di overflow viene posto uguale ad 1.

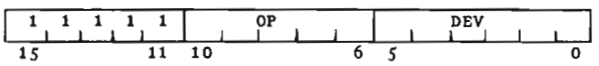
Per il registro Carry o Extended il codice è STE (Set Extended).
A seguito di tale istruzione il contenuto del registro di extend viene posto uguale ad 1.

Il campo N dell'istruzione contiene tutti zero.
La codificazione simbolica non operazione è NOP e le azioni da essa deter-

minate sono:

- il contenuto del registro C viene incrementato di 1
- non viene eseguita alcuna altra operazione.

Nel caso del calcolatore LABEN 70 il formato delle istruzioni di ingresso-uscita si presenta con la codificazione seguente:



L'istruzione è divisa in 3 campi.
Il campo costituito dai bit 15-11 è fisso e caratteristico di questo gruppo di istruzioni.
Il campo OP (bit 10-6) contiene il codice operativo.
Il campo DEV (bit 5-0), nel caso di istruzioni di ingresso-uscita, specifica il numero di identificazione della periferica selezionata; nel caso dell'istruzione di arresto non viene preso in esame.

Questa classe di istruzioni di ingresso-uscita può essere suddivisa nei seguenti sottogruppi:

- istruzioni per il trasferimento di dati;
- istruzioni per il trasferimento di comandi;
- istruzioni per il controllo delle periferiche;
- istruzioni per la simulazione di un canale di accesso diretto alla memoria.

Istruzioni per il trasferimento di dati.

Queste istruzioni permettono di trasferire dati da una periferica all'unità centrale (registro A o memoria) e viceversa attraverso il registro dati della periferica.
Le istruzioni appartenenti a questo gruppo vengono riportate in seguito.

Campo OP	Nome simbolico	Significato
01010	INA	Ingresso dati nel registro A
01000	INA	Ingresso dati nel registro A senza azzeramento del flag (specificato nel campo DEV)
00010	INM	Ingresso dati in una locazione di memoria
00000	INM	Ingresso dati in una locazione di memoria senza azzeramento del flag (specificato in DEV)
01110	OTA	Uscita dati dal registro A
01100	OTA	Uscita dati dal registro A senza azzeramento del flag (specificato in DEV)
00110	OTM	Uscita dati da una locazione di memoria
00100	OTM	Uscita dati da una locazione di memoria senza azzeramento del flag (specificato in DEV)

Esaminiamo ora le azioni determinate da tali istruzioni:

INA – Il contenuto del registro dati relativo all'unità di ingresso selezionata dall'operando viene trasferito nel registro A.

INA 2 : il contenuto del registro dati della periferica 2 viene trasferito nel registro A. Dopo aver eseguito il trasferimento il bistabile di flag viene azzerato.

INA 2,F : dopo il trasferimento, il bistabile di flag non viene azzerato.

OTA – Il contenuto del registro A viene trasferito nel registro dati relativo all'unità periferica selezionata dall'operando.

INM – Tale istruzione occupa due locazioni di memoria. Il contenuto del registro dati dell'unità periferica selezionata dall'operando viene trasferito nella locazione di memoria, il cui indirizzo è specificato dalla proposizione immeditamente seguente la INM.

INM 2 Il contenuto del registro dati della periferica 2 viene trasferito nella
DSA PIPPO locazione di memoria di indirizzo simbolico PIPPO.

OTM – Tale istruzione occupa due locazioni di memoria. Il contenuto della locazione di memoria, il cui indirizzo è specificato dalla proposizione immediatamente successiva alla OTM, viene trasferito nel registro dati relativo all'unità periferica selezionata dallo operando.

Istruzioni per il trasferimento di comandi.

Queste istruzioni sono impiegate per caricare il registro di comando delle periferiche e per leggere il contenuto del registro di stato delle periferiche. Le istruzioni appartenenti a questo gruppo vengono riportate in seguito.

Campo OP	Nome simbolico	Significato
01011	RSA	Trasferimento del contenuto del registro di stato nel registro A
01001	RSA	Trasferimento del contenuto del registro di stato nel registro A senza azzeramento del flag (specificato in DEV)
00011	RSM	Trasferimento del contenuto del registro di stato in una locazione di memoria
00001	RSM	Trasferimento del contenuto del registro di stato in una locazione di memoria senza azzeramento del flag
01111	WCA	Trasferimento del contenuto del registro A nel registro di comando
01101	WCA	Trasferimento del contenuto del registro A nel registro di comando senza azzeramento del flag
00111	WCM	Trasferimento del contenuto di una locazione di memoria nel registro di comando
00101	WCM	Trasferimento del contenuto di una locazione di memoria nel registro di comando senza azzeramento del flag

Esaminiamo le azioni determinate da tali istruzioni:

RSA – Il contenuto del registro di stato della periferica selezionata dall'operando viene trasferito nel registro A. L'esecuzione di tale istruzione determina l'azzeramento del flag e di alcuni indicatori della parola di stato, il numero e la natura dei quali varia a seconda del tipo di periferica.
Se però viene precisato il sottocampo opzionale F, il flag e tutti gli indicatori della parola di stato **non** vengono modificati.

RSM – Il contenuto del registro di stato della periferica selezionata dall'operando viene trasferito nella locazione di memoria, il cui indirizzo è precisato dalla proposizione immediatamente seguente la RSM.

WCA – Il contenuto del registro A viene trasferito nel registro di comando della periferica selezionata dall'operando. L'esecuzione di tale istruzione determina l'azzeramento del flag. Se però viene precisato il sottocampo F il flag **non** viene azzerato.
Ricordando quanto detto a proposito del registro di comando, tale istruzione consente di stabilire le modalità di funzionamento della periferica selezionata.

WCM – Il contenuto della locazione di memoria, il cui indirizzo è specificato dalla proposizione immediatamente seguente la WCM, viene trasferito nel registro di comando della periferica selezionata dall'operando.

Le istruzioni RSM e WCM sono, in generale, seguite da una pseudo-istruzione DSA che specifica l'indirizzo della locazione di memoria interessata.

Istruzioni per il controllo delle periferiche.

Queste istruzioni permettono di inviare all'unità periferica selezionata dall'operando particolari comandi, relativi al suo funzionamento a controllo di programma, ad interruzione automatica e ad accesso diretto alla memoria.
Le istruzioni appartenenti a questo gruppo vengono riportate nella seguente tabella:

Campo OP	Nome simbolico	Significato
10100	CLF	Azzeramento del flag
10110	STF	Posizionamento del flag
10000	SKF	Salto condizionato dal flag
10010	CLC	Azzeramento del bistabile di controllo
10011	STC	Posizionamento del bistabile di controllo
10001	SKC	Salto condizionato dal bistabile di controllo
11000	EIS	Abilitazione del sistema di interruzione
11000	DIS	Disabilitazione del sistema di interruzione
11010	DSC	Azzeramento di tutti i bistabili di controllo
10101	EMA	Abilitazione al funzionamento in accesso diretto alla memoria

Esaminiamo le azioni determinate da tali istruzioni:

CLF – Il flag dell'unità periferica selezionata dall'operando viene azzerato. La periferica porrà il flag uguale a 1, quando sarà pronta per il trasferimento di un nuovo dato.

STF – Il flag dell'unità periferica selezionata dall'operando viene messo a 1.

SKF – Se il flag dell'unità periferica selezionata dall'operando è uguale 1 non viene eseguita l'istruzione successiva; altrimenti si procede in sequenza. Questa istruzione merita un commento più particolareggiato.
Quando si ha a che fare con trasferimenti tra periferiche ed elaboratore un problema è presente, cioè la grande differenza esistente tra la velocità di elaborazione dell'unità centrale e la velocità della maggior parte delle periferiche. Ad esempio, considerando la telescrivente, il tempo necessario per l'acquisizione di un dato è di 100 ms., mentre si ricorda che un ciclo macchina è di 1,3 μs.
E' quindi necessario, nel funzionamento a controllo di programma, sincronizzare l'attività dell'elaboratore con la disponibilità della periferica; in altre parole è compito del programma controllare, prima di realizzare un trasferimento, che la periferica sia disponibile per il trasferimento stesso. Una periferica d'ingresso deve quindi poter segnalare di aver acquisito il dato e di essere pronta ad inviarlo all'unità centrale; un'unità di uscita deve invece poter segnalare di essere pronta ad acquisire il dato elaborato dall'unità centrale.

CLC – Azzera il bistabile di controllo dell'unità periferica selezionata dall'operando.
Ricordando il significato del bistabile di controllo, a seguito dell'istruzione in esame la periferica selezionata dall'operando viene disabilitata al funzionamento ad interruzione automatica.

STC – Il bistabile di controllo della periferica selezionata dall'operando viene messo a 1. La periferica viene in tal modo abilitata al funzionamento ad interruzione automatica.

SKC – Se il bistabile di controllo della periferica selezionata dall'operando è uguale a 1 non viene eseguita l'istruzione successiva, altrimenti si procede in sequenza.

EIS – Il sistema di interruzione automatica viene abilitato. Tale istruzione non altera la situazione dei bistabili di controllo delle periferiche.

DIS – Il sistema di interruzione automatica viene disabilitato. Tale istruzione non altera la situazione dei bistabili di controllo delle periferiche.

DSC – I bistabili di controllo di tutte le unità periferiche vengono azzerati. Il sistema di interruzione automatica viene disabilitato.

EMA – Abilita la periferica selezionata dall'operando ad operare su un canale di accesso diretto alla memoria. Si rimanda al funzionamento in accesso diretto alla memoria.

Istruzione per la simulazione di un canale di accesso diretto alla memoria.

Queste istruzioni consentono il trasferimento ad alta velocità di tabelle di dati da (o in) memoria. Permettono inoltre di eseguire operazioni aritmetiche sui dati trasferiti o sugli indirizzi che ad essi competono. Le istruzioni appartenenti a tale gruppo sono riportate nella seguente Tabella.

Campo OP	Nome simbolico	Significato
11100	ISM	Ingresso sequenziale in memoria
11101	OSM	Uscita sequenziale dalla memoria
11111	ADM	Addizione sequenziale di dati al contenuto della memoria
11110	ICM	Incremento in memoria

ISM – L'istruzione è costituita da 3 parole.
La prima parola contiene nel campo OP il codice operativo e nel campo DEV il numero di identificazione della periferica selezionata.
La seconda parola, indicata nel seguto con WC, contiene il numero (negativo) delle parole da trasferire. Essa viene utilizzata come contatore.
La terza parola, indicata nel seguito con ADDR, contiene un numero che viene usato come indirizzo di una locazione di memoria.

	ISM	a
WC	DAV	.....
ADDR	DSA	.....

Il contenuto del registro dati della periferica selezionata a viene trasferito nella locazione di memoria il cui indirizzo è contenuto in ADDR. Dopo il trasferimento i contenuti di WC e di ADDR vengono incrementati di 1 unità. In tal modo, una serie di dati vengono memorizzati in successive locazioni di memoria.
Se il contenuto di WC diventa uguale a zero, il trasferimento di memoria ha termine e viene messo a 1 un opportuno bit della parola di stato della periferica a.

OSM – L'istruzione è costituita da 3 parole.
La prima parola contiene nel campo OP il codice operativo e nel campo DEV il numero di identificazione della periferica selezionata.
La seconda parola, indicata nel seguito con WC, contiene il numero (negativo) delle parole da trasferire. Essa viene utilizzata come contatore.
La terza parola, indicata nel seguito con ADDR, contiene un numero, che viene usato come indirizzo di una locazione di memoria.

	OSM	a
WC	DAV	.....
ADDR	DSA	.....

Il contenuto della locazione di memoria il cui indirizzo è contenuto in ADDR, viene trasferito nel registro dati della periferica selezionata a. Dopo il trasferimento i contenuti di WC e di ADDR vengono incrementati di 1 unità. In tal modo è possibile inviare alla periferica una serie di dati memorizzati in successive locazioni di memoria. Se il contenuto di WC diventa uguale a zero il trasferimento viene arrestato, viene messo uguale a 1 un opportuno bit della parola di stato della periferica a e viene generata una interruzione di segnalazione, la cui locazione di arrivo è legata al numero di identificazione della periferica selezionata dalla formula vista ($400_8 + 4 \cdot N + 3$).

ADM – L'istruzione è costituita da 3 parole.
La prima parola contiene nel campo OP il codice operativo e nel campo DEV il numero di identificazione della periferica selezionata.
La seconda parola, indicata nel seguito con WC, contiene il numero (negativo) delle parole da trasferire. Essa viene utilizzata come contatore.
La terza parola, indicata nel seguito con ADDR, contiene un numero che viene usato come indirizzo di una locazione di memoria.

	ADM	a
WC	DAV	.....
ADDR	DSA	.....

Il contenuto del registro dati della periferica selezionata a è sommato al contenuto della locazione di memoria il cui indirizzo è contenuto in ADDR. Dopo la somma i contenuti di WC e di ADDR vengono incrementati di una unità. Il trasferimento ha termine quando il contenuto di WC diventa uguale a zero: viene messo uguale a 1 un opportuno bit della parola di stato della periferica a e viene generata una interruzione di segnalazione.

ICM – L'istruzione è costituita da 2 parole.
La prima parola contiene nel campo OP il codice operativo e nel campo DEV il numero di identificazione della periferica selezionata.
La seconda parola, indicata nel seguito con COST, contiene un numero che viene usato come una costante.

```
                ICM     a
COST            DAV     .........
```

Il contenuto del registro dati della periferica selezionata a viene sommato alla costante contenuta in COST. Il risultato di questa somma costituisce l'indirizzo della locazione di memoria interessata. Il contenuto della locazione così individuata viene incrementato di 1 unità. Se tale incremento determina un supero di capacità aritmetica (overflow) viene generata una interruzione di segnalazione.

Istruzione di arresto

Campo OP	Nome simbolico	Significato
10111	HLT	Arresto

Questa istruzione arresta il calcolatore; se poi viene premuto il pulsante RUN della console, l'esecuzione del programma viene ripresa a partire dall'istruzione immediatamente seguente la istruzione HLT.

4.3. LINGUAGGIO ASSEMBLATORE.

L'insieme delle istruzioni espresse nel codice binario, come descritto nei precedenti paragrafi, è detto "**linguaggio macchina**". Un programma scritto in termini di queste istruzioni è perciò un programma scritto in linguaggio macchina.
Scrivere un programma in questo linguaggio è compito non solo faticoso ma anche noioso e soggetto a facili errori per le seguenti considerazioni:

- tutte le istruzioni devono essere espresse in forma numerica codificata e non è facile ricordare a memoria o disporre con immediatezza il codice di ogni istruzione necessaria;
- ogni indirizzo deve essere definito con il proprio codice, in forma diretta; sicché il programmatore è costretto fin dall'inizio, quando ancora non può valutare con esattezza l'istruzione del programma che si

accinge a scrivere, a stabilire e fissare le locazioni dove memorizzare ogni dato ed ogni istruzione;
- cambiamenti di programma e di dati, come pure l'inserimento e la cancellazione di istruzioni o di sottoprogrammi, impone di ridefinire gli indirizzi delle istruzioni e dei dati;
- infine parti di programma e sottoprogramma che potrebbero essere applicati in generale in altri casi, non sono utilizzabili con generalità essendo necessario riassegnare gli indirizzi per adeguarsi di volta in volta alla struttura del nuovo programma.

Per queste ragioni sono stati sviluppati linguaggi che rappresentano la stessa informazione espressa in codice binario in una forma mnemonica più facile a capire ed a ricordare. E' evidente infatti che un termine mnemonico come "ADD" e "SOMMA" la cui dizione richiama l'operazione fatta eseguire dall' istruzione, è di uso più immediato del codice binario corrispondente alla stessa istruzione. La rappresentazione mnemonica può essere estesa oltre che alle istruzioni anche ai dati, consentendo così di scrivere un programma non solo senza dover specificare i codici delle istruzioni, ma anche senza dover fissare gli indirizzi in termini diretti assoluti.
L'insieme di questi termini mnemonici costituisce un linguaggio detto "**assemblatore**".
Nel programmare i primi calcolatori, fin da allora i tecnici scrivevano i programmi prima in forma mnemonica, cioè in linguaggio assemblatore, e poi traducevano ogni termine simbolico nel codice corrispondente del linguaggio macchina che solo può essere interpretato dall'unità di controllo del calcolatore. Ben presto però ci si accorse che proprio questa operazione di traduzione è un compito tipico che può essere affidato allo stesso calcolatore. Si sviluppò così per ogni tipo di calcolatore il cosiddetto "**programma assemblatore**", cioè un programma che effettua la traduzione dei programmi dal "**linguaggio assemblatore**" al "**linguaggio macchina**".

Il **programma assemblatore è quindi un traduttore**.
Ne consegue che per elaborare un programma usando il linguaggio assemblatore, occorre procedere attraverso le seguenti fasi:

1 – Si scrive il programma nel linguaggio assemblatore a cui si dà il nome di "**programma sorgente**";
2 – utilizzando il programma assemblatore si sottopone il programma sorgente alla traduzione automatica da parte dello stesso calcolatore, ottenendo così il programma scritto in linguaggio macchina a cui si

dà il nome di "**programma oggetto**";

3 – il programma oggetto viene quindi introdotto nel calcolatore per l'esecuzione.

Per quanto riguarda le fasi 2 e 3 è bene tener presente che gli assemblatori possono essere distinti in due categorie:

- quelli associati a macchine con memoria di lavoro di notevole dimensione che, durante lo stesso processo di traduzione, provvedono anche a memorizzare il programma oggetto già nelle locazioni di memoria indicate. Sono gli assemblatori detti: load-and-go (carica e va);
- nelle macchine con memoria di dimensione più modesta questa, essendo già occupata in buona parte proprio dal programma assemblatore, non può contenere anche il programma oggetto. Occorre perciò raccogliere quest'ultimo su un supporto fisico (nastro, schede, disco, ecc.) dopo di che si procede al suo caricamento nella memoria di lavoro, liberata dal programma assemblatore, e quindi alla sua esecuzione.

4.3.1. Le proposizioni del linguaggio assemblatore.

L'espressione significativa base di ogni linguaggio di programmazione, e quindi anche del linguaggio assemblatore, è una linea di termini mnemonici simbolici detta: "**proposizione**".

Definire uno specifico linguaggio di programmazione significa fissare prima i termini simbolici e quindi stabilire un insieme di regole che indicano come detti termini possono essere combinati insieme per formare le proposizioni del linguaggio.

Normalmente nei linguaggi assemblatori si distinguono tre classi di proposizioni come segue:

- proposizioni con istruzioni di comando macchina;
- proposizioni con dati;
- proposizioni con istruzioni di comando assemblatore, dette anche pseudoistruzioni e che danno luogo a pseudo-operazioni.

Le proposizioni di comando macchina sono quelle che il programma assemblatore traduce direttamente nelle corrispondenti istruzioni in codice di linguaggio macchina.
Le proposizioni con dati sono usate per memorizzare specifici valori o da-

ti in una determinata locazione di memoria.
Infine le proposizioni di comando assemblatore sono quelle che **non** conducono alla generazione di alcun codice in linguaggio macchina. Esse infatti sono usate per dare al programma assemblatore le informazioni su come si deve svolgere il processo di traduzione, ad esempio: quale è la locazione da cui occorre far iniziare la memorizzazione del programma oggetto; dove termina tale programma; quando termina il processo di traduzione e così via. Questo scambio d'informazioni è realizzato mediante le cosiddette **pseudoistruzioni** che dirigono il processo di traduzione del programma assemblatore e che, come le istruzioni di macchina, sono individuate nel linguaggio simbolico assemblatore da simboli mnemonici: questi termini simbolici non trovano però corrispondenza in un codice del linguaggio macchina, donde il nome di pseudoistruzioni.

4.3.2. Formato delle proposizioni del linguaggio assemblatore.

In generale una proposizione nel linguaggio assemblatore è costituita da quattro campi denominati:

- campo etichetta (label);
- campo codice operativo;
- campo variabile;
- campo commento.

I campi sono separati tra loro da uno o più spazi e si susseguono nell'ordine come segue:

campo etichetta	campo codice operativo	campo variabile	campo commento

Il campo etichetta serve per individuare una proposizione e può essere riempito con un simbolo di caratteri alfanumerici.
Sicché una stessa etichetta può essere utilizzata una sola volta, cioè non vi possono essere due o più proposizioni con la stessa etichetta.
Nel campo codice operativo, separato dal precedente da almeno uno spazio, sono specificate in termini mneomonici le istruzioni e le pseudoistruzioni.
Il campo variabile, separato anch'esso dal precedente da uno o più spazi, è riempito da un termine simbolico o da un dato numerico il cui formato

TABELLA 4.1 – CARATTERI DEL CODICE ASCII–8 BIT – TELETYPE

Caratteri	Codice ASCII (in ottale)	Caratteri	Codice ASCII (in ottale)
0	060	blank	040
1	061	≠	043
2	062	$	044
3	063	&	046
4	064	'	047
5	065	(	050
6	066	)	051
7	067	*	052
8	070	+	053
9	071	,	054
A	101	–	055
B	102	.	056
C	103	/	057
D	104	<	074
E	105	=	075
F	106	>	076
G	107	∩	100
H	110	[	133
I	111	]	135
J	112		
K	113	Print Enable	001
L	114	Print Suppress	004
M	115	Line Feed	012
N	116	Carriage Return	015
O	117	Reader on (X-ON)	021
P	120	Tape on (T-ON)	022
Q	121	Reader off(X-OFF)	023
R	122	Tape off(T-OFF)	024
S	123	RUB OUT	177
T	124		
U	125	Per la telescrivente OLIVETTI TE 300 si hanno le seguenti variazioni:	
V	126		
W	127		
X	130		
Y	131	Print Enable	017
Z	132	Print Suppress	016

e significato dipendono dal codice operativo della proposizione.
Il campo commento è opzionale e viene usato dal programmatore per annotare richiami esplicativi: i commenti infatti vengono ignorati dall'assemblatore.
Poiché nel processo di traduzione l'assemblatore traduce una proposizione per volta, è necessario indicare la fine della proposizione stessa con un termine simbolico costituito da una sequenza caratteristica di caratteri come ad esempio la seguente:

CR (carriage return) LF (line feed) OFF

4.3.3. Caratteri usati per formare i termini simbolici del linguaggio assemblatore.

Ogni termine simbolico del linguaggio assemblatore viene formato da una sequenza di caratteri scelti da un particolare insieme di caratteri che in generale coincidono con quelli del codice ASCII – 8 bit (Tabella 4.1).

4.3.4. Considerazioni sull'uso dei linguaggi assemblatori.

Da quanto è stato prima descritto, si osserva che i linguaggi assemblatori hanno le seguenti caratteristiche:

- sono specifici per un dato calcolatore e senza pratico significato per altri tipi di calcolatori;
- hanno una corrispondenza 1:1 fra le proposizioni del linguaggio e le istruzioni di macchina;
- per il motivo precedente, offrono la possibilità in modo semplice di rivelare errori di programmazione;
- il programma traduttore è compatto e tutti i programmi espressi nel linguaggio simbolico dell'assemblatore risultano più efficienti in termini di tempo di esecuzione e di spazio di memoria occupato, rispetto ai programmi espressi in linguaggio di livello più elevato (FORTRAN, BASIC, ecc.).

Come contropartita questi linguaggi presentano i seguenti svantaggi:

- i programmi non sono disponibili per uso generale, perché specifici per il tipo di calcolatore;
- una sintassi troppo elementare fa risultare legali sequenze ed azioni che invece possono essere senza significato;

- i codici troppo specifici e la scrittura di un gran numero di proposizioni in codici anche simbolici ma in un linguaggio non orientato al problema dà occasione di compiere numerosi sbagli anche se banali e rende i programmi di difficile lettura;
- infine l'utilizzatore deve impiegare molto tempo e fatica per prendere confidenza con le particolari caratteristiche sia del linguaggio che del calcolatore, prima di poter programmare in modo efficiente.

4.4. LINGUAGGI DI PROGRAMMAZIONE.

Le limitazioni prima segnalate nei linguaggi assemblatori hanno spinto, fin dal periodo iniziale dell'uso dei calcolatori, a sviluppare linguaggi la cui espressione e sintassi fossero orientate verso il problema da risolvere piuttosto che essere vincolate alla particolare struttura del calcolatore. Essi vengono detti **linguaggi di programmazione** o **linguaggi orientati verso le procedure e le applicazioni.**
Un linguaggio di questo tipo deve essere però sufficientemente flessibile da potersi usare per programmare una vasta classe di problemi. Questo si ottiene individuando prima le operazioni fondamentali di tipo aritmetico, logico e procedurale della gamma di problemi da trattare e quindi facendo in modo che queste operazioni costituiscano le proposizioni fondamentali del linguaggio: così ogni espressione risulta indipendente dai codici del particolare calcolatore usato.
La costruzione di questi linguaggi è perciò tale che essi possono essere adoperati senza richiedere che l'utilizzatore conosca come il calcolatore è strutturato per eseguire le diverse operazioni. Sicché il programmatore può concentrarsi su come risolvere meglio il suo problema piuttosto che preoccuparsi su come scrivere il programma da svolgere nella maniera più efficace dal calcolatore. Il compito di tradurre il programma,dal linguaggio orientato ai problemi al linguaggio macchina,è lasciato allo stesso calcolatore che compie questa operazione mediante un apposito programma di traduzione.
Vi sono essenzialmente due tipi di programmi di traduzione: il primo detto "compilatore" è un programma traduttore che trasforma un **intero** programma scritto in linguaggio sorgente – **programma sorgente** – in un corrispondente **programma oggetto** scritto in linguaggio macchina; il secondo tipo di traduttore detto "**interprete**" trasforma uno o più proposizioni del

programma sorgente direttamente nelle corrispondenti istruzioni di macchina e le esegue prima di procedere all'interpretazione delle successive, sicché interpreta ed esegue le proposizioni del programma sorgente senza produrre alcun programma oggetto.
A tutt'oggi si possono contare molti linguaggi di programmazione a diversi livelli di generalizzazione (*) ma i più noti sono:

- FORTRAN (FORmula-TRANslation) introdotto dalla IBM fin dallo inizio degli anni 50;
- ALGOL (ALGorithmic-Language) introdotto come standard internazionale, ma diffuso limitatamente in USA, mentre viene molto usato nei centri di ricerca e nelle università, particolarmente in Europa;
- PL/1 (Programming-Language-1) introdotto più recentemente per superare alcune limitazioni del FORTRAN e fornire ulteriori prestazioni non incluse nei precedenti linguaggi. E' prevedibile che questo linguaggio avrà una diffusione internazionale sempre maggiore;
- BASIC (Beginners All-purpose Simbolic Instruction Code) originalmente sviluppato al Dartmouth College (Hanover, New Hampshire U. S.A.) dai Professori J. Kemeny e T.E. Kurrt come linguaggio conversazionale da utilizzare nei terminali in time-sharing ed oggi molto diffuso per la sua estrema semplicità in tutti gli impieghi dei sistemi interattivi uomo-macchina.

Altri linguaggi come COBOL (Common-Business-Oriented-Language) oppure LISP, SNOBOL, IPL-V, sono anch'essi abbastanza noti ma risultano specializzati perché orientati verso problemi amministrativi, processi di apprendimento, di simulazione o per la manipolazione dei simboli.

Come si intuisce, nessuno dei linguaggi menzionati è universale nel senso che può trattare ogni classe di problemi: essi non hanno una struttura così estesa come quella dei linguaggi naturali e vengono perciò detti anche linguaggi artificiali.

4.4.1. Programmi compilatori.

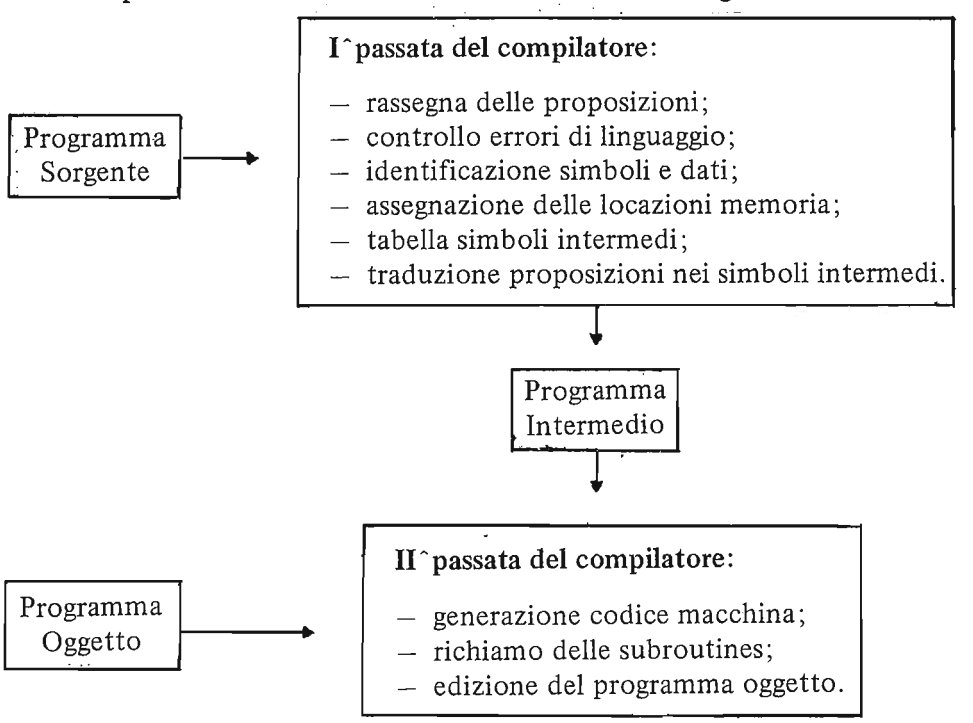
Un programma scritto in un linguaggio generalizzato orientato verso il problema, deve essere tradotto nel linguaggio macchina proprio del calcolatore impiegato per poter essere eseguito automaticamente: a questa tradu-

(*) Sammet J.E.: "Programming Languages: history and fundamentals" – Prentice-Hall, N.J.

zione può provvedere, come già si è detto, un programma generale detto "**compilatore**".
Tale "compilatore" farà passare automaticamente dal particolare programma **sorgente** scritto nel linguaggio generalizzato, al corrispondente programma **oggetto** riscritto nel linguaggio macchina che il calcolatore sarà così in grado di eseguire.
I minicomputers risultano in generale corredati con il compilatore FORTRAN e solo in casi eccezionali anche con quello ALGOL o PL/1.
Per quanto riguarda il BASIC, essi sono in generale corredati, come vedremo in seguito, con il programma di traduzione interprete e solo in alcuni casi dispongono anche di programma di traduzione da compilatore.

4.4.2. Come agisce un compilatore.

Sotto la direzione del **compilatore** il calcolatore elabora le espressioni del programma sorgente in modo da tradurne parole e simboli nel proprio codice operativo. Idealmente questo processo di traduzione potrebbe essere ottenuto anche in un solo ciclo di lettura o passata del programma sorgente. In pratica esso viene portato a termine in due o più passate.
Per descrivere la procedura generale seguita da un compilatore a due passate ci si può riferire allo schema a blocchi mostrato in fig. 4.1



Durante la prima passata il compilatore esplora ed analizza tutte le proposizioni del programma nel linguaggio sorgente, classifica ogni espressione e rivela gli eventuali errori formali commessi nella scrittura del programma sorgente. Identifica le variabili, i dati e le istruzioni ed assegna loro le locazioni della memoria. Costruisce la tabella dei simboli relativa alle variabili del programma sorgente.
Alla fine di questa prima fase viene in qualche modo prodotto un programma intermedio in cui le informazioni vengono organizzate e classificate in funzione della struttura del compilatore e delle caratteristiche della macchina usata.
Nella seconda fase il compilatore opera nuovamente una scansione delle proposizioni del linguaggio sorgente e sulla base delle informazioni prodotte durante la prima passata produce il programma oggetto espresso in codici di macchina.

4.4.3. Considerazioni sull'uso dei linguaggi compilatori.

Il FORTRAN è il linguaggio di programmazione orientato verso i problemi di ingegneria, più adoperato nell'impiego con i minicomputers. Vi sono due versioni compatibili con le specifiche ASA (American Standard Association):

- FORTRAN II detto anche ASA-Basic;
- FORTRAN IV detto anche ASA-Standard.

Le due versioni sono compatibili, essendo il FORTRAN II un sottoinsieme del IV. Normalmente i minicomputers sono corredati non solo del compilatore FORTRAN IV ma anche di una libreria di subroutines che contiene le più comuni funzioni matematiche (logaritmo, seno, coseno, ecc.).
Anche l'ALGOL è un linguaggio generalizzato per impiego nei problemi scientifici e di ingegneria normalmente usato nello standard ALGOL-60 (vedasi ALGOL-60 Report – Communication of the ACM – January 1963). Esso adotta una sintassi meglio definita più estesa e più facile ad apprendersi che per il FORTRAN, ma è deficitario per quanto riguarda la gestione generalizzata dell'input-output che dipende dal tipo della macchina usata. Il compilatore ALGOL è in genere molto più esteso e richiede più memoria che il compilatore FORTRAN. E' per questo motivo che i minicomputers in generale, se sono corredati anche per la traduzione in

ALGOL, accettano solo un sottoinsieme più limitato delle istruzioni ALGOL.
In generale si può dire che la codificazione di un problema nel linguaggio orientato al problema non è così efficiente in termini di memoria occupata e tempo di esecuzione come quella ottenibile scrivendo direttamente il programma in linguaggio macchina od in assemblatore. Però come contropartita ciò consente di:

- accedere più facilmente al calcolatore da parte di utenti anche non esperti ed introdotti sulla struttura interna della macchina usata;
- scrivere più rapidamente i programmi esprimendoli in una forma più immediatamente comprensibile alla lettura dell'uomo;
- disporre di programmi riutilizzabili su calcolatori diversi perché scritti in un linguaggio legato al problema e svincolato dalla macchina.

4.4.4. Programmi interpreti (BASIC).

Per i linguaggi di programmazione orientati verso il problema, sono stati sviluppati anche i cosiddetti programmi interpreti, che differiscono dai compilatori essenzialmente per il modo con cui viene effettuata la traduzione nel linguaggio-macchina.
Abbiamo visto che un compilatore traduce in una o più passate ed una volta per tutte il programma sorgente nel programma oggetto che è poi disponibile, scritto nel linguaggio macchina, per essere eseguito una, due o più volte dal particolare calcolatore usato.
Un programma interprete è invece caratterizzato dal fatto che esso esegue direttamente il programma scritto nel linguaggio sorgente senza produrre un equivalente programma oggetto.
Infatti esso interpreta una frase alla volta in modo che ad ogni proposizione corrisponda una determinata sequenza di istruzioni nel codice macchina, che viene subito mandata in esecuzione.
Un interprete **non genera mai un programma oggetto completo** il quale possa essere riutilizzato all'occorrenza in seguito, perché interpretazione ed esecuzione delle proposizioni sono continuamente inframmezzate. Il programma viene perciò eseguito già nella sua espressione in linguaggio sorgente.
Ne consegue che, mentre i compilatori sono particolarmente indicati in tutti quei casi in cui i programmi devono essere reimpiegati più volte, gli interpreti sono di grande vantaggio nei casi in cui il programma deve esse-

re eseguito una o poche volte e quando si richiede di conoscere risultati intermedi di calcolo mano mano che la esecuzione del programma procede nelle successive fasi.
E' questo il caso in cui il calcolatore viene utilizzato per compiere, ad esempio, operazioni matematiche alla stregua di una macchina calcolatrice veloce, oppure quando è impiegato come terminale in time-sharing che può interrogare un centro di calcolo in modo sempre diverso per eseguire calcoli numerici.
Il linguaggio che ha avuto larghissima diffusione in questo senso è il BASIC. Oggi quasi tutti i minicomputers sono corredati con il programma **interprete BASIC** perché il corrispondente linguaggio di programmazione è relativamente semplice da imparare e da usare.
Le istruzioni BASIC somigliano da vicino al linguaggio parlato e le regole sintattiche del linguaggio da imparare a memoria sono poche e semplici. Ogni proposizione nel BASIC occupa una linea caratterizzata all'inizio da un numero come etichetta distintiva; l'interprete BASIC è tale che automaticamente le proposizioni vengono ordinate secondo i numeri di etichetta crescenti.
L'edizione del programma è fatta molto semplicemente sia riscrivendo sulla tastiera di una telescrivente ogni proposizione sbagliata nella forma corretta, sia assegnando un numero di linea fra le due linee entro cui una nuova proposizione va inserita: a questo proposito i numeri di etichetta fra linee successive vengono distanziati ogni volta di una decade in modo da avere nove numeri intermedi per possibili future correzioni o nuove inserzioni di linee.
Il BASIC è anche detto linguaggio conversazionale perché, nei sistemi interattivi uomo-macchina, il programma rimane in memoria nella forma di programma "**sorgente**" così come è stato scritto dall'utente ed ogni esecuzione o chiamata di informazioni viene svolta e presentata direttamente nella forma BASIC giacché il programma interprete svolge, come si è detto, la traduzione proposizione per proposizione.
Questo fatto conduce anche ad un altro vantaggio non secondario del BASIC nei sistemi interattivi, dovuto al fatto che il programma da eseguire viene scritto ed edito in una sola forma ed in una sola copia (sorgente) senza dover procedere alla traduzione ed alla edizione completa in programma oggetto come richiesto nel caso di un compilatore.
Per altro proprio queste caratteristiche mentre rendono il BASIC estremamente apprezzato nelle applicazioni in time-sharing, in quelle di calcolo

matematico ed in molti usi di laboratorio, ne limitano viceversa i vantaggi nelle applicazioni per il controllo dei processi in tempo reale proprio a causa del maggior tempo di elaborazione richiesto in conseguenza della continua e ripetuta operazione di traduzione richiesta al programma interprete.

4.5. SOFTWARE DI SISTEMA.

Il corredo software di un calcolatore comprende, oltre all'assemblatore ed ai programmi compilatori ed interpreti dei linguaggi adottati, anche i programmi del cosiddetto "**software di sistema**".
Questo è costituito dall'insieme dei programmi che consentono al calcolatore di operare in modo automatico nella preparazione e nella esecuzione dei programmi da elaborare e nella gestione delle varie unità di cui è composto un sistema. Sicché i programmi del software di sistema vengono spesso classificati nelle due grandi categorie seguenti:

- **programmi di servizio** (utilities o utility programs) che sono un insieme di programmi destinati a far eseguire le fasi di rilocazione e di caricamento (linking loader), di edizione (editor), di correzione (debugging) ed altre analoghe (diagnostic, input-output routines) dei programmi in elaborazione nel calcolatore;
- **sistemi operativi** (operating systems o executives) che sono programmi complessi destinati a governare compiti e funzioni svolte dalle varie unità del sistema ed a regolarne il flusso dei dati durante lo svolgimento del lavoro. I sistemi operativi gestiscono perciò non solo le unità hardware del sistema, ma anche il corredo software e quindi i programmi di servizio disponibili i quali, a questo riguardo, vanno perciò considerati come dei sottoprogrammi del più generale programma adottato come **sistema operativo**. Compito fondamentale del sistema operativo è quindi quello di gestire in modo automatico le risorse hardware e software del sistema per l'esecuzione dei programmi secondo una propria strategia.

Tra i programmi di servizio possono essere elencati e definiti i seguenti di cui sono forniti tutti i calcolatori sia che abbiano oppure no un sistema operativo che li comprenda come sottoprogrammi:

4.5.1. Linking Loader.

Questo programma è di fondamentale importanza per un calcolatore ed in particolare per i mini calcolatori. Questi infatti hanno in generale modalità di indirizzamento che prevedono la definizione di indirizzi assoluti. Per questa ragione un programma in linguaggio macchina non può essere eseguito in una qualsiasi posizione in memoria ma solo nella zona cui si riferiscono gli indirizzi assoluti generati.
Né il programma assemblatore, né il programma compilatore sono in grado di produrre un programma oggetto con indirizzi assoluti, soprattutto quando si desideri collegare al programma eventuali sottoprogrammi di utente o della libreria di sistema. I programmi traduttori sono in grado infatti di produrre solo indirizzi relativi all'inizio del programma, ma non quelli relativi alle aree comuni (COMMON) e ai sottoprogrammi da collegare al programma principale. Essi producono quindi in realtà un programma oggetto rilocabile che non può essere eseguito prima che gli indirizzi vengano rilocati in base ad una posizione assegnata nella memoria del calcolatore e prima che vengano prelevati dal supporto fisico (nastro di carta, nastro magnetico, disco) su cui sono memorizzati i sottoprogrammi di libreria, tutti i sottoprogrammi necessari alla esecuzione del programma principale.
Queste operazioni di rilocazione e di collegamento e caricamento di sottoprogrammi (linkage editing) sono eseguite dal "Linking Loader" sul programma oggetto prodotto dai programmi traduttori, producendo un programma assoluto in codice macchina effettivamente eseguibile.

4.5.2. Loader.

E' un breve programma progettato in modo che quando viene immagazzinato in memoria permette al calcolatore di accettare ed immagazzinare altri programmi. Vi possono essere vari tipi di loader a seconda del tipo di linguaggio in cui il programma da caricare è scritto ed a seconda dell'unità di ingresso da cui il programma è fornito.

4.5.3. Editor.

E' un programma destinato a generare i programmi in edizione su schede, su nastro di carta o su nastro magnetico eliminando il lavoro di doverlo fare in modo manuale. Quando un programma sorgente è stato battuto su telescrivente ed immagazzinato in memoria, esso può essere collaudato,

corretto o modificato. Quindi, allorché si è pronti per l'edizione finale, su semplice comando dato dall'operatore, l'Editor può generare il programma su nastro o su altro supporto in forma adatta per essere elaborata da programmi assemblatori o compilatori così da ottenere i programmi oggetto eseguibili dal calcolatore.

4.5.4. Debugging.

E' un programma destinato ad assistere il programmatore per trovare gli errori commessi nello stendere i programmi. Quando un programmatore ritiene che il suo programma è corretto e pronto per essere usato, egli può caricarlo in memoria e farlo scorrere fino alla fine (se arriva ad una fine!). Se il programma non fornisce i risultati corretti o non si ferma affatto continuando a ripetere cicli di istruzioni errati, il programmatore può usare i comandi di consolle della CPU per esaminare una alla volta le locazioni di memoria in cui il programma è memorizzato e tentare di trovare gli errori eventuali attraverso l'interpretazione delle informazioni presentate sul display del calcolatore. E' evidente come questo procedimento passo-passo in modo manuale diventi prolisso e faticoso. Lo stesso procedimento può essere svolto con il programma di debugging al quale su comando esterno può essere affidato il controllo delle fasi di elaborazioni. In questo modo il programma evolve fino ad un punto desiderato in cui l'elaborazione si ferma al fine di consentire al programmatore di esaminare ed eventualmente modificare contenuti di registri, dati ed istruzioni. L'utente può così interrompere l'operazione del suo programma in qualsiasi punto ed esaminare se lo stato del programma e del calcolatore è quello previsto, in modo da isolare e correggere le cause di errori.

4.5.5. Diagnostic.

E' un programma destinato a controllare il corretto funzionamento della CPU, delle unità di memoria e delle periferiche. Esso è utilizzato in caso di guasti del sistema per individuarne la causa oppure viene impiegato nelle procedure di manutenzione periodica.

4.5.6. Monitor.

In generale questo non è un programma indipendente, ma quasi una estensione software dell'hardware del sistema. Dopo l'iniziale caricamento, il monitor risiede permanentemente nella memoria interna per controllare,

qualunque sia il programma in esecuzione, il flusso dei dati di input-output, la protezione di memoria, le procedure di interruzione.
Il suo compito è quello di conservare una descrizione dei vari programmi durante la loro esecuzione, di destinare tempo di calcolo fra programmi concorrenti e di attuare le procedure che i programmi possono richiamare per interagire fra loro.
Il programma monitor viene attivato su comando di interruzione e può eseguire una sequenza di istruzioni come entità indivisibili e in modo privilegiato rispetto agli altri programmi.

4.5.7. Sistemi operativi.

Quando il sistema diventa complesso perché formato oltre che dalla CPU anche da diversi tipi di memoria, come unità a disco, registratori a nastro magnetico, e da numerose periferiche di ingresso ed uscita, la preparazione e la gestione dei programmi possono diventare molto laboriose.
Per semplificare i compiti del programmatore, sono stati sviluppati i cosiddetti "sistemi operativi". Ad esempio un sistema operativo per unità a disco o per unità a nastro, dovrà gestire in modo automatico tutti i trasferimenti di dati fra CPU e la memoria di massa, dovrà consentire che tutti i programmi standard risiedano nella memoria di massa quando non sono in esecuzione ma siano automaticamente richiamabili e disponibili quando necessari.
In questo modo tutto il corredo software può facilmente essere utilizzato dal programmatore senza la necessità per lui di caricare i programmi ogni volta prima di ogni traduzione o elaborazione.
Oltre ai programmi DOS (Disc Operating System) e MTOS (Magnetic Tape Operating System) ne sono stati sviluppati diversi altri specializzati in funzione dei compiti e della composizione del sistema. Così il "Batch Processing Operating System" è usato per organizzare l'esecuzione a lotti di diversi programmi simultanei; mentre il "Time-Sharing Operating System" è impiegato per organizzare gli ingressi da diversi terminali di entrata e destinare ad ognuno una quota del tempo della CPU.

Infine il più generale sistema definito come "Real Time Multiprogramming Operating System" è progettato per gestire simultaneamente più programmi e dispone di programmi per organizzare le chiamate di interruzione, per schedare un insieme di diversi programmi e per gestire tutti i trasferimenti di dati e di programmi.

Il termine "Real Time" deriva dal fatto che il sistema deve essere in grado di rispondere, cioè di fornire uscite entro un tempo richiesto e determinato da processi esterni che si svolgono indipendentemente dal lavoro del calcolatore.
Da tutto quanto si è accennato, si desume che un sistema operativo è perciò costituito da un insieme di programmi compatibili per interagire fra loro in modo da regolare il lavoro del sistema e da gestire automaticamente ed efficientemente **tutte le risorse** del sistema stesso.
Quando si confrontano diversi sistemi operativi in tempo reale (Real-Time Operating System), la scelta va definita tenendo presente:

- le strategie adottate per destinare le disponibilità di memorie;
- le strategie per destinare i tempi di calcolo e le priorità di interruzione;
- la gestione dell'Input-Output;
- le opzioni per etichettare e schedare altre funzioni;
- le disponibilità di rivelare e correggere errori, di sovraintendere e regolare lo svolgimento dei programmi.

I sistemi operativi offrono diversi grandi vantaggi come: semplificazione nello scrivere i programmi applicativi; disponibilità di tutti i programmi di servizio utilizzabili in modo compatibile; rapidità nella realizzazione di un sistema di calcolo; documentazione meno copiosa e nello stesso tempo più significativa.
A fronte di tutto questo si paga però sia nei tempi di risposta alla interruzioni software ed hardware che possono diventare più lunghi rispetto al caso di programmi specializzati e non generalizzati, sia nelle richieste di maggiori risorse del sistema in termini di interfacce hardware e software e di capacità di memoria.

Capitolo 5

LA MICROPROGRAMMAZIONE, LA MICROELETTRONICA E LE STRUTTURE FIRMWARE.

5.1. GENERALITA'.

Nel 1951 il Prof. Wilkes, inaugurando una conferenza sui calcolatori all'Università di Manchester , coniò per primo il termine "microprogrammazione" e ne descrisse i principi di base. L'obiettivo era quello di ricercare un metodo più sistematico e semplice per progettare il controllo di una macchina di Von Neuman. La sua idea base suggeriva di procedere al progetto dell'unità di controllo di un calcolatore non attraverso la definizione di un "sistema sequenziale utilizzante reti logiche e bistabili" ma adottando viceversa una memoria utilizzata in sola lettura, in cui venissero memorizzate parole binarie – dette microparole – con il codice corrispondente ai comandi da dare al calcolatore affinché esso assumesse lo stato desiderato. Sicché i segnali di comando per attuare i successivi stati del calcolatore venivano ottenuti leggendo in successione ordinata le microparole della "**memoria di controllo**" piuttosto che fornendo, con le istruzioni di macchina, segnali logici ad un sistema sequenziale il quale evolveva nelle successive configurazioni logiche relative ai diversi stati.
A quell'epoca l'interesse della proposta risiedeva soprattutto nella semplificazione del lavoro di progetto, nella possibilità di modularizzare le strutture e di normalizzare le procedure di diagnostica e di manutenzione dei calcolatori. Ma a parte alcune significative realizzazioni di calcolatori che tra il 1951 e il 1965 adottarono questo principio, tra cui anche il calcolatore CEP progettato presso l'Università di Pisa , la proposta rimase

senza grande seguito sul piano delle realizzazioni industriali. Ciò può essere giustificato dal fatto che in tale periodo le tecnologie ed i componenti elettronici, anche se già si prestavano al progetto di Wilkes, non consentivano tuttavia di raggiungere costi competitivi e risparmi rispetto ai progetti con reti logiche sequenziali.
Negli anni 60 e soprattutto con l'inizio degli anni 70, da una parte la comprensione sempre più estesa ed approfondita dei processi di elaborazione dei dati e dall'altra lo sviluppo di nuove tecnologie elettroniche costruttive hanno reso le applicazioni delle memorie di controllo in sola lettura economicamente e tecnicamente vantaggiose. Questi vantaggi sono diventati ancor più evidenti con l'avvento dei componenti integrati a larga scala con i quali l'economia di un sistema di elaborazione è sempre più determinata dai costi di progetto, di manutenzione e di gestione piuttosto che dal costo dei componenti elementari formanti i singoli circuiti logici. Infatti gli stessi circuiti integrati hanno un "layout" orientato non dal numero di componenti ma dalla funzione operativa richiesta nella manipolazione dei segnali. Sicché la microelettronica ha contribuito a far risorgere un vasto interesse applicativo nella "microprogrammazione".
Gli studi e le applicazioni si sono così diversificati che oggi è difficile trovare una definizione univoca e universalmente accettata per tutto ciò che va sotto il nome di microprogrammazione.
Infatti essa oggi investe svariati aspetti della scienza dei calcolatori oltre a quello messo in luce inizialmente da Wilkes sul progetto delle unità di controllo. Nella sistemistica, ad esempio, la microprogrammazione è vista come una tecnica molto efficace per interpretare con un calcolatore il funzionamento parziale o totale di un altro calcolatore di struttura diversa: tale situazione è comunemente definita come "emulazione" o "simulazione".
Nell'architettura dei sistemi essa può rendere compatibili fra loro calcolatori diversi, consentendo così una normalizzazione delle procedure di gestione. Nel software è possibile realizzare direttamente in forma "hardware" microprogrammi speciali e di "routine", il che conduce a linguaggi di programmazione di livello più elevato e di tipo conversazionale. Infine la microprogrammazione incide profondamente anche nel progetto della strumentazione specializzata e dei cosiddetti "calcolatori dedicati ad una speciale applicazione" (dedicated digital machines): le funzioni che queste macchine devono svolgere possono essere progettate come un "programma software", realizzate in "hardware" con una memoria di controllo micro-

programmata dove restano permanentemente conservate e possono essere richiamate durante il funzionamento dello strumento con semplici codici simbolici generati da pulsanti a pannello.
Poiché in tutte le applicazioni che sono state brevemente accennate, si utilizza una memoria per generare i segnali di comando necessari a svolgere le funzioni richieste, possiamo perciò convenire in senso generale di indicare con "microprogrammazione" la tecnica di generare i segnali di comando mediante memorizzazione in una "memoria di controllo". Questa tecnica viene anche detta "firmware" per indicare che la sequenza dei comandi costituisce un patrimonio "fisso" della macchina, che viene studiato e progettato come un "programma software" e realizzato con strutture "hardware".

5.2. COMPITI DELL'UNITA' DI CONTROLLO DI UN CALCOLATORE.

Abbiamo già visto nel secondo capitolo che, fra le varie sottounità funzionali della unità centrale di calcolo CPU, l'unità di controllo è quella da cui dipende in gran parte il funzionamento dell'intero sistema giacché essa fornisce i segnali di temporizzazione e di comando per tutte le operazioni da effettuare.
Per mettere meglio in evidenza questa funzione dell'unità di controllo, si riporta in fig. 5.1 lo schema a blocchi della CPU in forma leggermente modificata rispetto allo schema già analizzato nel capitolo secondo, in modo da sottolineare come i segnali di uscita dall'unità di controllo costituiscano i comandi di entrata dei circuiti logici per aprire e chiudere nei tempi desiderati il flusso dei dati fra le diverse sottounità della CPU.
Ogni operazione elementare – trasferimento dati, operazione logica o aritmetica, memorizzazione dati, ecc. – è completamente determinata da un insieme di segnali di comando, ossia da un insieme o stringa di bits generati dall'unità di controllo per i circuiti logici che regolano il flusso dei dati. Questi comandi vanno quindi visti come i segnali che attuano gli "scambi" per organizzare le vie percorse dai dati e che controllano i cicli di temporizzazione, cioè determinano il tempo in cui deve avvenire il passaggio del dato.
Per rendere meno vaga questa descrizione si esamini lo schema di fig. 5.2: in questo esempio, ognuno dei tre registri A, B e C può essere commutato sull'ingresso dell'Unità Aritmetica; l'uscita di questa a sua volta può esse-

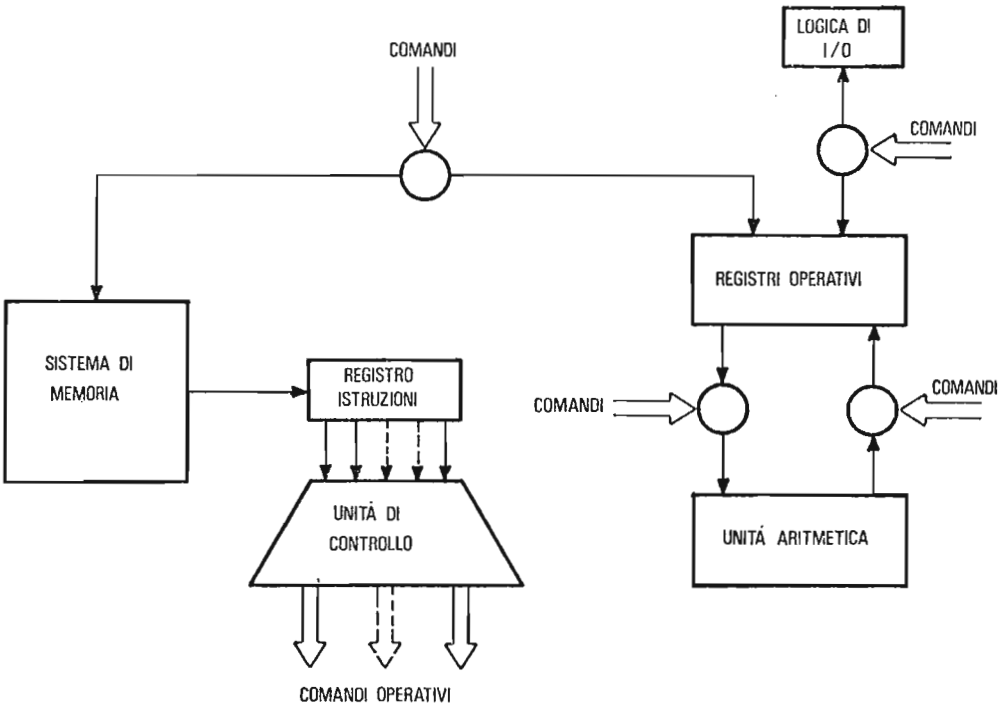


FIG. 5.1 – SCHEMI A BLOCCHI DI UN ELABORATORE DI DATI PRESENTATO PER METTERE IN EVIDENZA LA FUNZIONE DELL'UNITÀ DI CONTROLLO CHE COMANDA IL FLUSSO DEI DATI.

re commutata sull'ingresso di un qualsiasi registro.
E' evidente che per compiere un'operazione, ad esempio sommare i contenuti dei registri A e B depositare il risultato nel registro C, è necessario applicare segnali di comando sia alle porte logiche che determinano gli accoppiamenti per i desiderati flussi di dati, sia all'unità aritmetica perché compia l'operazione richiesta. Inoltre poiché per commutare i circuiti logici e per consentire il trasferimento dei dati fra i diversi circuiti occorre un tempo finito, questi segnali di comando devono presentarsi all'istante opportuno e durare per un periodo determinato in modo da consentire la effettuazione completa dell'operazione. Questo periodo di tempo è quindi preso a riferimento come "intervallo elementare o ciclo temporale" per le commutazioni del flusso dei dati e per le operazioni di trasformazione dei dati. I segnali di scansione di questi cicli sono forniti dalla stessa unità di controllo.

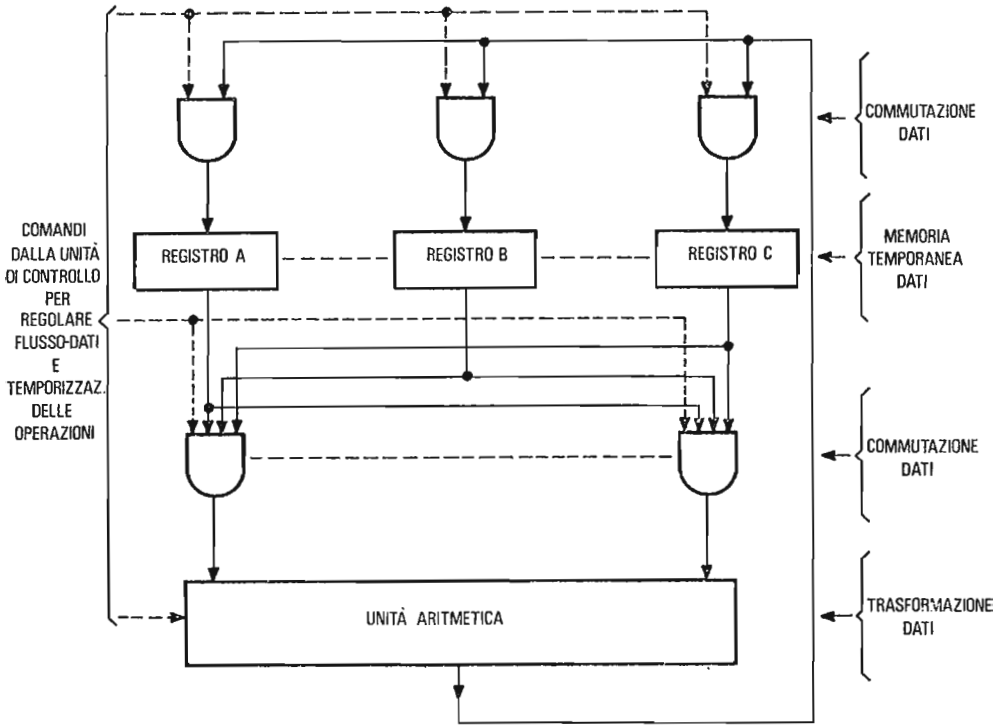


FIG. 5.2 – SCHEMA A BLOCCHI DI PRINCIPIO PER EVIDENZIARE LA FUNZIONE DEI SEGNALI "COMANDO OPERAZIONE" FORNITI DALLA UNITÀ DI CONTROLLO DELLA CPU.

5.3. L'UNITA' DI CONTROLLO CON RETI SEQUENZIALI.

Le unità di controllo per fornire i segnali di comando e di temporizzazione così descritti sono state realizzate negli anni passati soprattutto con reti logiche sequenziali le quali, controllate dagli impulsi di un orologio elettronico, evolvono in successione temporale da uno stato ad un altro in funzione sia dello stato precedente che dei segnali forniti da altre condizioni esterne.

Una generica schematizzazione a blocchi di queste reti può essere quella riportata in fig. 5.3. L'insieme è formato da un certo numero di bistabili

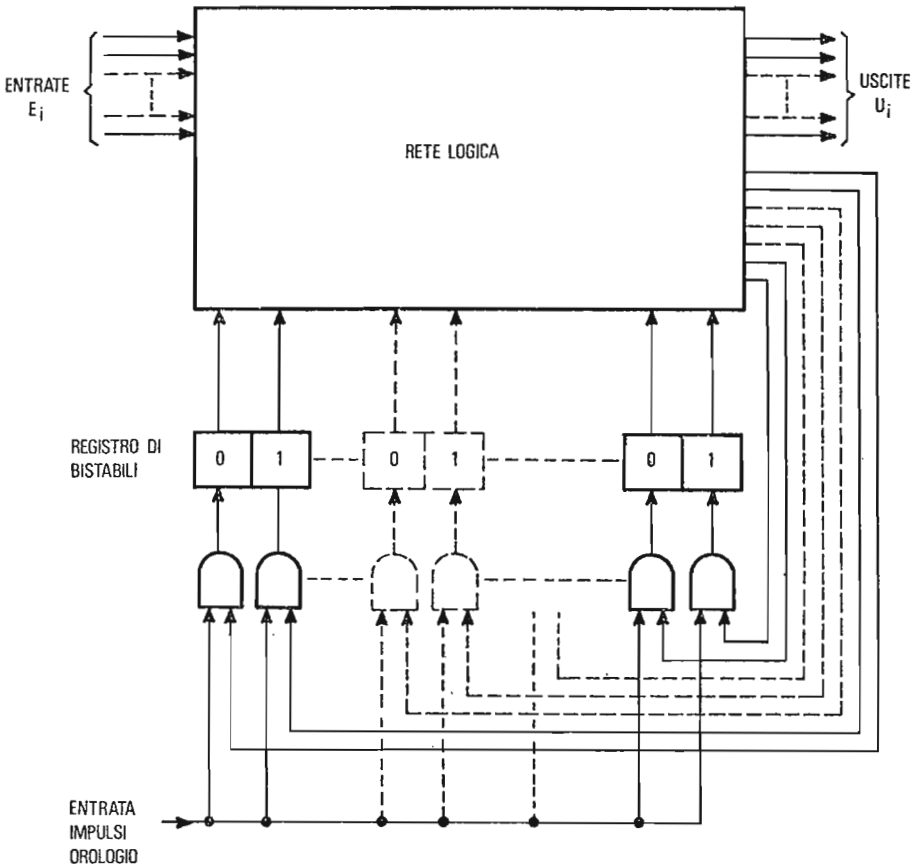


FIG. 5.3 – SCHEMA DI PRINCIPIO DI UNA RETE LOGICA SEQUENZIALE.

(ad esempio del tipo R-S come in Figura), le cui uscite sono fra loro collegate da una rete logica che determina la successione degli stati.
In una rete comprendente più bistabili lo stato logico dell'intera rete può essere descritto in termini degli stati in cui si trovano i singoli elementi bistabili: si suole anche chiamare "superstato" della rete ogni stato logico definito dall'insieme degli stati dei suoi bistabili. Per una rete con n bistabili si potranno avere 2^n stati o superstati diversi. Il superstato che esiste al tempo $t = n+1$ è una funzione:

1) del superstato al tempo $t = n$;
2) degli impulsi esterni ricevuti dalla rete al tempo $t = n$;
3) della struttura della rete logica che collega i diversi bistabili.

In ogni istante la configurazione degli stati assunta dagli elementi bistabili può essere rappresentata con un codice binario che individua il superstato della rete.
Per ogni superstato si ha una determinata configurazione di uscite della rete logica che sono in 1 oppure in 0.
Queste uscite selezionano quindi gli AND di ingresso che consentiranno il comando in SET o in RESET dei bistabili al prossimo impulso di orologio onde ottenere il passaggio al superstato che segue nella sequenza desiderata. Le entrate ausiliarie della rete logica possono essere usate per variare a piacere l'ordine di successione dei superstati, cioè per passare da una sequenza all'altra.
Si possono così avere diversi tipi di sequenze dei 2^n superstati. Si possono inoltre prevedere sequenze con un numero più limitato $m < 2^n$ di successivi superstati differenti.
La transizione fra un superstato ed il successivo può essere compiuta in modo "sincrono" oppure "asincrono".
Nelle reti di bistabili operanti in modo sincrono si utilizzano gli impulsi di orologio, in corrispondenza dei quali si determinano le transizioni fra i superstati.
Nelle reti operanti in modo sincrono, gli stati si evolvono solo in corrispondenza di una variazione degli ingressi e non esistono segnali di orologio.
Questo tipo di soluzione, sebbene può far raggiungere la maggior economia nel numero dei componenti elettronici e dei circuiti logici impiegati, comporta però numerose complicazioni nel definire la struttura organizzativa del calcolatore, nell'adozione delle metodologie di progetto modulare, ed infine nell'individuare procedure standard di collaudo, di diagnosi di guasti e di manutenzione. Infatti i blocchi logici del sistema sequenziale dell'unità di controllo risultano alla fine divisibili solo concettualmente e non fisicamente dai blocchi delle altre unità della CPU.

Ne consegue non solo una difficoltà concettuale di prevedere tutte le configurazioni e le interferenze possibili fra i blocchi logici, ma anche una complessità circuitale che infine limita la velocità operativa a causa di lunghe catene di elementi logici in cascata, e che crea problemi per le difficol-

tà di messe a punto e per la eliminazione dei disturbi di interferenza.

Appare quindi naturale che fin dall'inizio dello sviluppo dei calcolatori si ponesse attenzione a metodologie alternative per il progetto delle unità di controllo della CPU. Non a caso il primo lavoro di Wilkes in questo campo si intitola "The best way to design an automatic calculating machine" e non per esempio "Microprogramming design philosophy".

5.4. L'UNITA' DI CONTROLLO MICROPROGRAMMATA SECONDO IL MODELLO DI WILKES.

Il concetto di unità di controllo con la sequenza dei comandi memorizzata in una memoria fissa non modificabile – memoria in sola lettura ROM – è stato introdotto da Wilkes notando che ciascuna delle istruzioni di macchina può essere scomposta in una sequenza programmata di più operazioni elementari con cui attivare le porte logiche interessate. Queste sequenze, dette anche microprogrammi, invece di essere generate da una rete logica sequenziale, come prima si è detto, vengono fornite da una struttura con memoria in sola lettura come quella riportata in fig. 5.4 che segue lo schema originale dell'autore.
Esso consiste di una matrice di controllo C, di una matrice di sequenza S, di un decodificatore e di un bistabile di condizione. Le matrici, nella soluzione inizialmente proposta, sono costituite da memorie a nuclei di ferrite in sola lettura in cui ogni nucleo posto all'incrocio fra una riga ed una colonna costituisce una cella logica "OR". Il decodificatore in base all'indirizzo fornito dal registro seleziona una linea orizzontale: ogni linea orizzontale può essere pensata come una microistruzione che una volta selezionata fornisce lungo le linee di uscita verticali segnali 1 negli incroci dove è posizionato un nucleo e segnali 0 negli altri incroci.
Le uscite verticali della matrice C costituiscono i segnali di comando delle porte logiche, mentre le uscite della matrice S costituiscono il codice dell'indirizzo della successiva microistruzione che viene fornito al registro con un opportuno ritardo. Il bistabile di condizione viene impiegato per decidere la selezione tra due possibili linee orizzontali in funzione di determinate situazioni verificatesi nel sistema, in modo da poter effettuare un salto o la scelta fra due potenziali successive istruzioni.

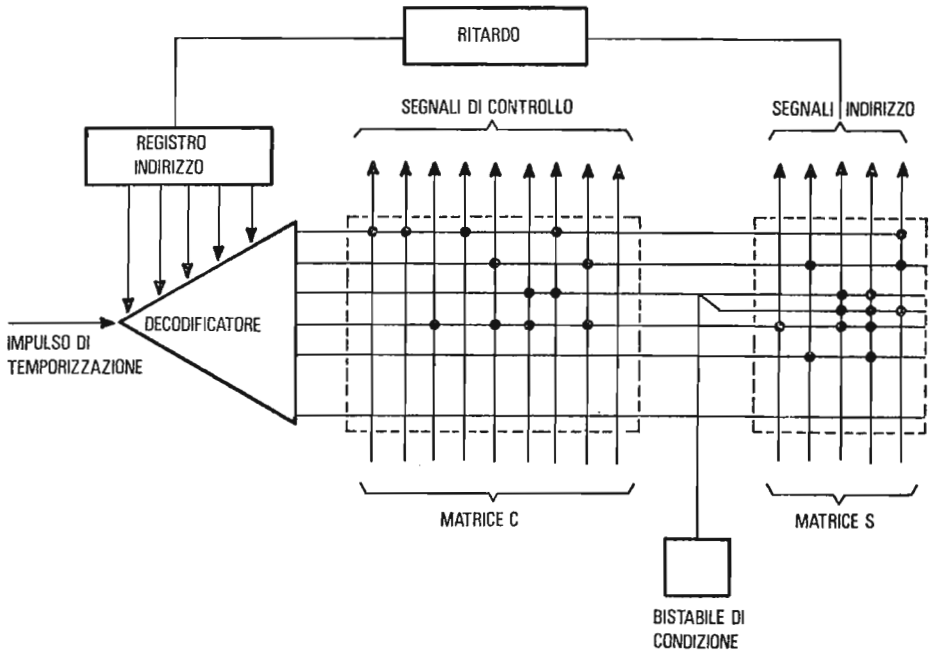


FIG. 5.4 – SCHEMA A BLOCCHI DEL MODELLO DI WILKES PER L'UNITÀ DI CONTROLLO MICROPROGRAMMATA.

Perciò riassumendo, il modello di Wilkes è caratterizzato dalle seguenti prestazioni:

- il numero delle linee orizzontali, selezionate dal decodificatore, rappresenta il numero delle microparole o microistruzioni memorizzate;
- il numero delle linee verticali uscenti sia dalla matrice C che da S, rappresenta il numero di bits di parola della microistruzione: in particolare il numero delle linee verticali di C costituiscono i segnali di comando per le reti logiche, quelle di S l'indirizzo della microistruzione successiva;
- il bistabile di condizione consente di variare la sequenza delle microistruzioni in funzione delle situazioni che si creano durante l'elaborazione.

5.5. CRITERI DI PROGETTO DELL'UNITA' DI CONTROLLO MICROPROGRAMMATA.

Abbiamo visto che le parti destinate a svolgere le operazioni importanti di un calcolatore sono composte dei seguenti elementi:

- elementi di memoria dati;
- elementi di commutazione e di trasferimento dati;
- elementi di trasformazione dati.

Questi elementi operano tutti sotto il governo dell'unità di controllo che regola il flusso dei dati e comanda l'esecuzione delle operazioni. L'unità di controllo che contiene l'informazione per il governo del sistema, può essere progettata sia come sistema sequenziale utilizzante solo cicuiti logici "OR", "AND", "NOT" e bistabili, sia come sistema con memoria in sola lettura ottenendo così una struttura più ordinata e regolare che viene detta "microprogrammata".
Analizziamo allora quali compiti devono essere affidati alla microistruzione memorizzata nella matrice della memoria di controllo. Essi sono essenzialmente relativi alle seguenti quattro funzioni:

– la **prima funzione** della microistruzione è quella di fornire i segnali per posizionare le linee di controllo logico durante ogni ciclo. La tecnica più semplice e banale sarebbe quella di assegnare nella microparola un bit per ogni linea di controllo: ma ovviamente questa soluzione è scartata per ragioni di efficienza perché vi sarebbero troppe combinazioni di valore dei bits disponibili che non verrebbero mai usate. Infatti se **n** sono le linee di controllo, al limite si potrebbero usare solo **m** bits di microparola, facendo $2^m = n$, e quindi utilizzare le **n** combinazioni degli **m** bits per comandare un decodificatore con **n** uscite. Tra queste due soluzioni estreme vi è tutta una varietà di possibilità suggerite dal tipo di sistema e di applicazione in istudio. In generale si suddividono i segnali di controllo in gruppi che sono logicamente in mutua esclusione, cioè che possono essere attivati uno alla volta in configurazioni significative associate per comandare le cosiddette "microoperazioni". Fra i gruppi di bits, che forniscono i cosiddetti segnali di "microcomando", e le linee di controllo si frappongono quindi i decodificatori di gruppo come mostrato in fig. 5.5.

– una **seconda funzione** della microistruzione è quella di determinare l'indirizzo della istruzione successiva. Si potrebbe pensare al metodo sempli-

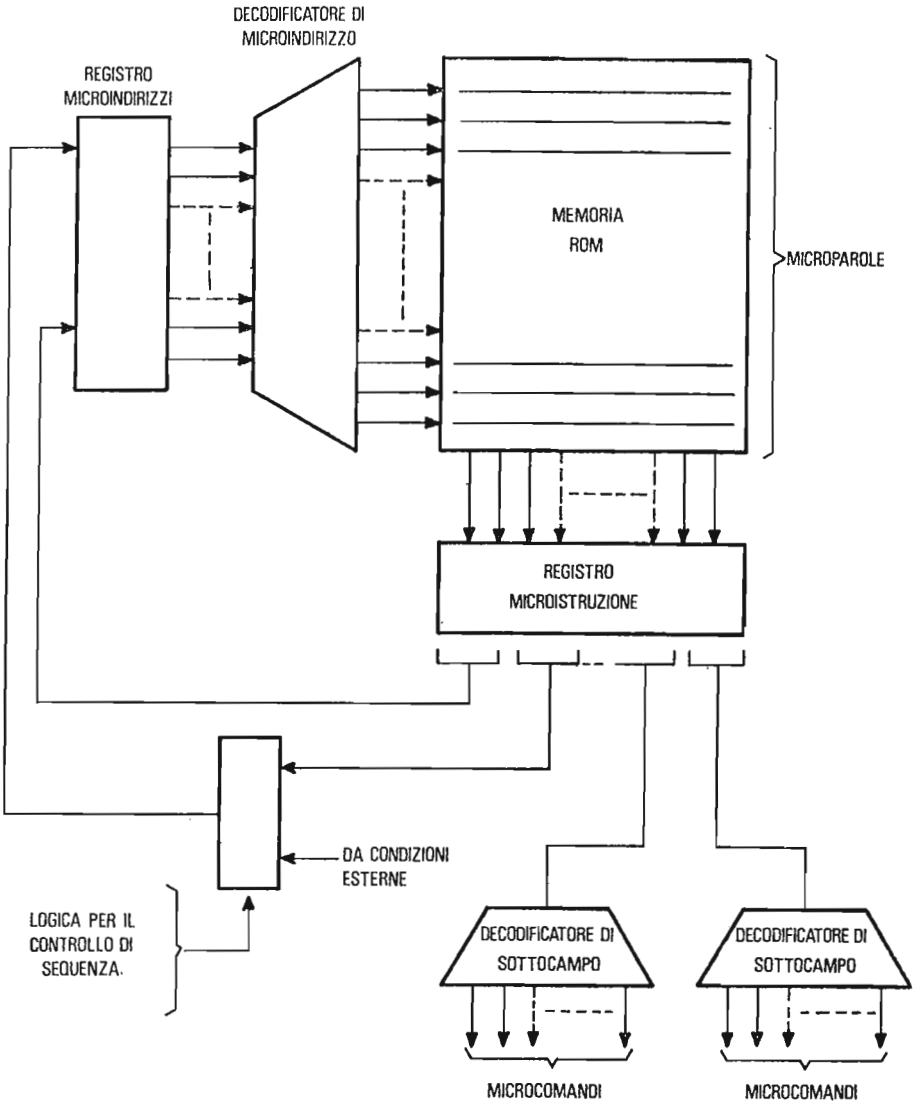


FIG. 5.5 – SCHEMA A BLOCCHI DI UNA UNITÀ DI CONTROLLO PROGRAMMATA.

ce di incrementare di una unità l'indirizzo dell'istruzione corrente. Questo ed altri metodi simili non sono tuttavia soddisfacenti perché non consentono di modificare la sequenza delle istruzioni in funzione delle condizioni che il sistema viene ad avere durante l'elaborazione dei dati: occorre cioè fornire il mezzo, come già si è visto nel modello di Wilkes, di variare

l'indirizzo in funzione sia dello stato precedente sia di condizioni esterne supplementari. Un metodo perciò oggi seguito è quello di riservare un campo di un certo numero di bits nella microparola che viene condizionato da situazioni esterne per modificare la sequenza, come mostrato in fig. 5.5.

– la **terza funzione**, altrettanto importante, della microistruzione è quella di controllare e comandare la durata delle microoperazioni. Questa funzione viene attuata specificando il ritardo con cui chiamare l'istruzione successiva, oppure stabilendo le condizioni da soddisfare prima di procedere oltre.

– infine una **quarta funzione** è quella di riservare dei bits per particolari attività di controllo, come ad esempio il controllo di parità, o per introdurre particolari valori numerici e costanti nel flusso dei dati.

Corrispondentemente nella microparola potremo avere quattro campi a cui sono affidate le funzioni anzidette. Alcuni comandi possono essere già impliciti nella logica delle sequenze microprogrammate: come ad esempio il fatto che l'indirizzo successivo alla istruzione corrente sia lo stesso incrementato di una unità a meno che non vi siano altri segnali di condizionamento esterno; oppure che la durata di esecuzione delle microistruzioni sia sempre la stessa a meno di altri condizionamenti.
Lo schema a blocchi di una unità di controllo microprogrammata si presenta allora come in fig. 5.5 in cui, sia l'inizio di sequenza del microprogramma, sia le modifiche di sequenza durante la sua esecuzione sono comandate da segnali esterni che provengono dal registro di istruzione principale della CPU, e da altre unità esterne.

5.6. LA MICROPROGRAMMAZIONE E LA SIMULAZIONE DI SISTEMI DI ELABORAZIONE.

La microprogrammazione è stata adottata già in forma molto estesa nei calcolatori della cosiddetta III^ generazione per progettare le unità di controllo della CPU. Oggi in diverse applicazioni dei minicomputers, la microprogrammazione viene impiegata anche per simulare sistemi di elaborazione.
Per spiegare cosa si vuole intendere come "simulazione", supponiamo di avere disponibile un sistema di calcolo A, detto **sistema ospite**. Su questo

sistema si voglia eseguire un programma già pronto e scritto per un altro sistema B.
Noi possiamo scrivere un insieme di programmi nel codice del sistema A che consentano di accettare in A i codici e quindi eseguire i programmi scritti per il sistema B.
La combinazione del sistema A e dell'insieme dei programmi di traduzione può essere considerato come un nuovo **sistema virtuale** che simula il sistema B e si comporta funzionalmente come B ad eccezione di quanto riguarda ovviamente i tempi di esecuzione delle elaborazioni.
L'idea di formare un sistema virtuale per rendere compatibili i programmi scritti per calcolatori diversi, può essere estesa in modo da interessare la struttura hardware interna del calcolatore. Abbiamo già visto che teoricamente è possibile distinguere le parti funzionali (memoria, trasferimento e trasformazione dati) dalla parte di controllo che regola il flusso dei dati e la sequenza delle istruzioni. Se questa seconda parte di controllo viene realizzata in hardware in modo da ripetere la logica di controllo e le istruzioni del sistema B, noi possiamo accoppiarla con le parti funzionali del sistema A in modo da attuare il sistema B simulato. In generale, questa possibilità integrale di sostituire l'unità di controllo del sistema A con una che simula quella di B non si verifica. Però adottando la tecnica della microprogrammazione è molto più agevole integrare l'unità di controllo A con memorie aggiuntive di controllo aventi opportuni microprogrammi, le quali si presentano fisicamente come unità aggiuntive del sistema A, ausiliarie per la simulazione di B con A.

Tutto questo che permette di raggiungere una migliore compatibilità fra diversi sistemi di calcolo, è oggi fattibile tecnicamente e giustificabile economicamente grazie al fatto che la microprogrammazione è facilmente attuabile con i circuiti integrati a larga scala LSI che contengono diverse decine di migliaia di elementi MOS come celle di memoria in sola lettura. E' anzi prevedibile che la tendenza a sostituire con la microprogrammazione in memorie di controllo (firmware) gran parte dei programmi e delle procedure "software" diventi sempre più diffusa in avvenire (hardwarizzazione del software) specialmente nelle attività ripetitive.

5.7. STRUTTURE "FIRMWARE" PER ALTRE DIVERSE APPLICAZIONI DELLA MICROPROGRAMMAZIONE.

Con l'inizio degli anni 70, proprio grazie ai criteri di progetti dei sistemi

microprogrammati con memorie di controllo da una parte, e dall'altra alle tecnologie dei componenti integrati LSI e delle memorie MOS in sola lettura (EPROM) sta trasformandosi la tradizionale netta suddivisione fra "hardware" e "software".
In passato infatti si è notato un continuo spostamento di funzioni dello hardware al software. Ciò mentre rendeva più economiche e nello stesso tempo potenzialmente più flessibili le unità fisiche del sistema di elaborazione, in realtà poneva condizionamenti invisibili sul piano della utilizzazione e della gestione dei sistemi a causa dell'enorme impegno richiesto nel lavoro di "software".
Oggi l'uso di memorie integrate a semiconduttore, ed i criteri di progetto delle unità microprogrammate, stanno diffondendo nelle applicazioni più diversificate la tecnica intermedia nota con il nome di "firmware" che rappresenta un metodo nuovo di ottimizzazione dei sistemi di calcolo. Con tale tecnica, come già si è detto con una frase sintetica ma molto espressiva, il sistema si progetta come "software" e si realizza come "hardware".
Già oggi le aree di applicazione del firmware nel progetto dei nuovi sistemi, sono le seguenti:

- controllo delle unità di ingresso e uscita; controllo nella gestione e nelle assegnazioni di memoria; gestione delle procedure di interruzione delle protezioni di memoria; unità di interfaccia e così via;
- realizzazioni in hardware del software applicativo più ricorrente e standardizzabile quali ad esempio le operazioni di lettura di tabelle di dati, le funzioni matematiche di uso generale, le funzioni specializzate ma ricorrenti in applicazioni specifiche;
- adattamento di sistemi per uso generale alle esigenze particolari di una installazione: lo stesso utente potrà essere messo in condizioni di progettare e modificare i microprogrammi mano mano che variano le condizioni di applicazione;
- infine la stessa struttura della strumentazione digitale di misura ed analisi viene trasformata poiché invece di procedere con il progetto delle reti sequenziali e con il conto dei circuiti logici, si adotta sempre di più la tecnica delle memorie di controllo con componenti integrati per attuare le operazioni specializzate a cui questa strumentazione è destinata. Sicché la nuova tecnica può condurre anche ad una maggiore compatibilità fra la strumentazione di misura ed i sistemi di elaborazione di dati.

INDICE

CAPITOLO 1 - EVOLUZIONE DEI SISTEMI DI ELABORAZIONE DI DATI
1.1 La tecnologia elettronica e le generazioni dei calcolatori. Pag. 5
1.2 La microelettronica e le prestazioni dei nuovi sistemi..... " 7
1.3 I criteri di progetto dei nuovi sistemi con i componenti della microelettronica..... " 9
1.4 Evoluzioni e tendenze dei sistemi di elaborazione di dati.. " 11

CAPITOLO 2 - STRUTTURE HARDWARE
2.1 Struttura hardware di un sistema minicomputer..... " 15
2.2 Struttura base della CPU in un minicomputer..... " 16
2.3 Strutture delle istruzioni di programma..... " 21
2.3.1 Le istruzioni di riferimento alla memoria..... " 22
2.3.2 Le istruzioni di riferimento in memoria ad un solo indirizzo nei minicomputers..... " 24
2.3.3 Istruzioni di operazione dei registri..... " 27
2.3.4 Istruzioni di comando per l'ingresso-uscita..... " 27
2.3.5 Istruzioni microprogrammate o microistruzioni..... " 28
2.4 L'unità di controllo e la temporizzazione delle fasi operative del calcolatore..... " 29
2.5 Il sistema di ingresso-uscita..... " 35
2.5.1 Modo su "Controllo di Programma"..... " 38
2.5.2 Modo su "Interruzione Automatica" (Interrupt)..... " 38
2.5.3 Modo di accesso diretto alla memoria DMA..... " 40

CAPITOLO 3 - SISTEMI DI MEMORIA
3.1 Generalità..... " 43
3.2 Le tecnologie delle unità di memoria..... " 44
3.3 Strutture delle unità di memoria..... " 47
3.4 Memorie statiche e dinamiche..... " 49
3.5 Memorie statiche ad accesso casuale: RAM..... " 50
3.6 Memorie statiche sequenziali..... " 50
3.7 Memorie ROM..... " 51
3.8 Memorie a nuclei magnetici..... " 53
3.8.1 Metodo di selezione 2D..... " 55
3.8.2 Metodo di selezione 3D..... " 57
3.8.3 Metodo di selezione 2½ D..... " 60
3.9 Le memorie a semiconduttore..... " 61
3.9.1 Generalità..... " 61
3.9.2 Le memorie RAM a semiconduttore..... " 62
3.9.3 Le memorie ROM a semiconduttore..... " 72
3.10 Le memorie a trasferimento di carica (CCD)..... " 76
3.10.1 Generalità..... " 76
3.10.2 Principio di funzionamento di una memoria a trasferimento di carica..... " 79

3.11 Le memorie a bolle magnetiche..... Pag. 82
3.11.1 Generalità..... " 82
3.11.2 Principio di funzionamento di una memoria a bolle magnetiche..... " 84
3.12 Memorie con superfici magnetiche..... " 91
3.12.1 Generalità..... " 91
3.12.2 I dischi magnetici..... " 92
3.12.3 I nastri magnetici..... " 94
3.12.4 Le mini unità di registrazione..... " 96

CAPITOLO 4 - STRUTTURE SOFTWARE
4.1 Generalità sulle tecniche di programmazione..... " 101
4.2 Lista delle istruzioni tipiche di un minicomputer..... " 105
4.2.1 Istruzioni di riferimento alla memoria..... " 106
4.2.2 Istruzione di operazione dei registri..... " 108
4.2.3 Istruzioni di ingresso-uscita..... " 108
4.3 Linguaggio assemblatore..... " 128
4.3.1 Le proposizioni del linguaggio assemblatore..... " 130
4.3.2 Formato delle proposizioni del linguaggio assemblatore... " 131
4.3.3 Caratteri usati per formare i termini simbolici del linguaggio assemblatore..... " 133
4.3.4 Considerazioni sull'uso dei linguaggi assemblatori.. " 133
4.4 Linguaggi di programmazione..... " 134
4.4.1 Programmi compilatori..... " 135
4.4.2 Come agisce un compilatore..... " 136
4.4.3 Considerazioni sull'uso dei linguaggi compilatori... " 137
4.4.4 Programmi interpreti (BASIC)..... " 138
4.5 Software di sistema..... " 140
4.5.1 Linking Loader..... " 141
4.5.2 Loader..... " 141
4.5.3 Editor..... " 141
4.5.4 Debugging..... " 142
4.5.5 Diagnostic..... " 142
4.5.6 Monitor..... " 142
4.5.7 Sistemi operativi..... " 143

CAPITOLO 5 - LA MICROPROGRAMMAZIONE , LA MICROELETTRONICA E LE STRUTTURE FIRMWARE
5.1 Generalità..... " 145
5.2 Compiti dell'unità di controllo di un calcolatore..... " 147
5.3 L'unità di controllo con reti sequenziali..... " 149
5.4 L'unità di controllo microprogrammata secondo il modello di Wilkes..... " 152
5.5 Criteri di progetto dell'unità di controllo microprogrammata..... " 154
5.6 La microprogrammazione e la simulazione di sistemi di elaborazione..... " 156
5.7 Strutture "firmware" per altre diverse applicazioni della microprogrammazione..... " 157

=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=

Prezzo L. 3.000